

# An Empirical Study of AI Techniques in Mobile Applications

Yinghua Li<sup>a</sup>, Xueqi Dang<sup>a,\*</sup>, Haoye Tian<sup>b</sup>, Tiezhu Sun<sup>a</sup>, Zhijie Wang<sup>c</sup>, Lei Ma<sup>c,d</sup>, Jacques Klein<sup>a</sup>  
and Tegawendé F. Bissyandé<sup>a</sup>

<sup>a</sup>University of Luxembourg, Luxembourg, Luxembourg

<sup>b</sup>The University of Melbourne, Melbourne, Australia

<sup>c</sup>University of Alberta, Edmonton, Canada

<sup>d</sup>University of Tokyo, Tokyo, Japan

## ARTICLE INFO

*Keywords:*

AI Apps

AI Technologies

Analysis

Empirical Study

## ABSTRACT

The integration of artificial intelligence (AI) into mobile applications has significantly transformed various domains, enhancing user experiences and providing personalized services through advanced machine learning (ML) and deep learning (DL) technologies. AI-driven mobile apps typically refer to applications that leverage ML/DL technologies to perform key tasks such as image recognition and natural language processing. Despite existing research exploring how mobile apps exploit AI techniques, they have the following main limitations: 1) Most existing studies focus on DL-based apps, with limited research on ML-based apps. 2) Existing research typically focuses on investigating the apps and the technologies utilized in the apps, lacking user-level analysis. 3) The number of apps studied is limited, with only 1,000 to 2,000 ML/DL apps identified after filtering. To fill the gap, in this paper, we conducted the most extensive empirical study on AI applications, exploring on-device ML apps, on-device DL apps, and AI service-supported (cloud-based) apps. Our study encompasses 56,682 real-world AI applications, focusing on three crucial perspectives: **1) Application analysis**, where we analyze the popularity of AI apps and investigate the update states of AI apps; **2) Framework and model analysis**, where we analyze AI framework usage and AI model protection; **3) User analysis**, where we examine user privacy protection and user review attitudes. Our study has strong implications for AI app developers, users, and AI R&D. On one hand, our findings highlight the growing trend of AI integration in mobile applications, demonstrating the widespread adoption of various AI frameworks and models. On the other hand, our findings emphasize the need for robust model protection to enhance app security. Additionally, our study highlights the importance of user privacy and presents user attitudes towards the AI technologies utilized in current AI apps. We provide our AI app dataset (currently the most extensive AI app dataset) as an open-source resource for future research on AI technologies utilized in mobile applications.

## 1. Introduction

The advent of artificial intelligence (AI) has revolutionized numerous fields, with mobile applications being a significant beneficiary. AI-driven mobile apps leverage machine learning algorithms, natural language processing, and computer vision to enhance user experiences, improve functionalities, and provide personalized services. These advancements are particularly evident in areas such as recommender systems (Lu, Wu, Mao, Wang and Zhang (2015); Wang, Wang and Yeung (2015)) handwriting recognition (Pham, Bluche, Kermorvant and Louradour (2014)) and face detection (Hjelmås and Low (2001); Dospinescu and Popa (2016)).


From the perspective of the underlying technology, AI applications can be broadly categorized into two main categories: machine learning-based apps (Sun, Sun, Lu and Mislove (2021)) and deep learning-based apps (Xu, Liu, Liu, Lin, Liu and Liu (2019)). Machine learning apps rely on traditional machine learning algorithms (e.g., decision trees (Rokach and Maimon (2005)), clustering algorithms (Xu

and Tian (2015)), and logistic regression (LaValley (2008))) to achieve the apps' functionality. For example, in the field of healthcare, logistic regression can be used for diabetes prediction (Joshi and Dhakal (2021)).

Deep learning apps leverage deep learning techniques to perform various tasks, such as image recognition (Jones (2020)), speech recognition (Matarneh, Maksymova, Lyashenko and Belova (2017)), and natural language processing (Locke, Bashall, Al-Adely, Moore, Wilson and Kitchen (2021)). For instance, Google Lens (Bilyk, Shapovalov, Shapovalov, Megalinska, Zhadan, Andruszkiewicz, Dołhańczuk-Śródka and Antonenko (2020)) is a prevalent deep learning mobile app that uses image recognition technology to identify objects, landmarks, and text in photos. Users can point their phone's camera at an object, and Google Lens can provide information about what they see, such as identifying the type of flower or offering detailed information about a landmark.

From the perspective of deployment methods, AI apps can be categorized into two groups: on-cloud inference and on-device inference (Xu et al. (2019)). In cloud-based inference, mobile devices connect to cloud servers via the network and transmit data to the servers. ML/DL models operating on the servers carry out inferences on the data and send the results back to mobile devices. Cloud-based inference benefits from robust computing capabilities and flexibility, making it suitable for dealing with intricate

\*Corresponding author.

 yinghuali@acm.org (Y. Li); xueqi.dang@uni.lu (X. Dang); haoye.tian@unimelb.edu.au (H. Tian); tiezhu.sun@uni.lu (T. Sun); zhijie.wang@ualberta.ca (Z. Wang); ma.lei@acm.org (L. Ma); jacques.klein@uni.lu (J. Klein); tegawende.bissyande@uni.lu (T.F. Bissyandé)

models and large data volumes. Nonetheless, it relies on network connectivity, which can lead to latency and data privacy concerns. In contrast, on-device inference enables ML/DL models to run directly on the mobile device. On-device inference offers low latency and offline capabilities, ensuring immediate responses and better data privacy protection. However, the drawback is that mobile devices have limited computing resources, posing challenges in handling complex models.

In the literature (Xu et al. (2019); Sun et al. (2021)), some studies have focused on investigating the deployment of ML/DL on mobile devices. Xu et al. (2019) present the first empirical study on how real-world Android apps exploit DL techniques. Their research focuses on three aspects: the characteristics of DL apps, what they use DL for, and what their DL models are. Sun et al. (2021) present the first empirical study of ML model protection on mobile devices. This study explored the extent of model protection usage in apps, the robustness of existing model protection techniques, and the potential impacts of stolen models.

Although their research is valuable and in-depth, there are the following limitations in the scope of their study.

- The majority of existing studies (Xu et al. (2019); Sun et al. (2021)) concentrated on DL-based apps, with a lack of research on classical ML-based apps. While existing studies (Sun et al. (2021)) explored model protection in ML apps, their focus has primarily been on the protection aspect without delving into the specific characteristics of ML apps or examining the utilization of ML technologies.
- Existing studies mainly focused on investigating the apps and the technologies utilized in the apps, lacking analysis at the user level. However, user reviews play a vital role in improving AI applications. Moreover, protecting user privacy in AI apps is crucial for maintaining user trust and ensuring compliance with data protection regulations. Existing studies lack such an analysis from the user's perspective.
- The number of apps studied in existing research is limited. Typically, the range of apps investigated in existing studies is between 10,000 to 50,000. Researchers then filter out the ML/DL apps. Finally, the number of ML/DL apps obtained from this process ranges from 1,000 to 2,000.

To fill this gap, we conducted the most extensive empirical study on AI applications, comprehensively exploring on-device machine learning (ML) apps, on-device deep learning (DL) apps, and AI service-supported apps (also referred to as cloud-based apps). Our study encompasses 56,682 real-world AI applications. To this end, we designed an automated AI app identification tool named AI Discriminator. On the server, AI Discriminator runs concurrently on 96 threads for approximately 1440 hours (two months), extracting 56,682 real-world AI applications from a pool of 7,259,232 mobile apps in the AndroZoo large-scale application repository (Allix, Bissyandé, Klein and Le Traon

(2016)). The number of AI apps we collected is currently the largest AI app dataset.

Specifically, our research focuses on three main perspectives: **1) Application analysis**, which includes AI app popularity analysis and AI app update status analysis. **2) Framework and model analysis**, which includes AI framework usage analysis and AI model protection analysis. **3) User analysis**, which includes user privacy protection analysis and user review analysis. Below, we provide a detailed explanation of each aspect of our empirical analysis.

- ① **AI app popularity analysis (Application analysis)** From the perspective of app popularity analysis, we investigate the following four aspects: **1) The annual development volume of AI apps** Understanding the annual development volume of AI apps can provide insight into the growth and adoption rate of AI technologies, contributing to identifying trends in AI investment. **2) The popular markets for AI apps** Identifying the popular markets for AI apps allows developers to understand where AI technology is being most widely adopted and integrated, providing insights into market entry strategies and investment decisions. **3) The prevalent categories of AI apps in the industry** Studying the prevalent types of AI apps in the industry highlights which categories of AI apps are most commonly developed and used. This information can inform developers and companies about the most in-demand AI functionalities, helping them align their development efforts with industry needs. **4) The popular categories of AI apps in the market** Understanding the popular categories of AI apps in the market provides a detailed view of consumer preferences. It helps identify which types of AI applications are gaining traction among users, providing developers insights for better-targeted product development and marketing strategies.
- ② **AI app update status analysis (Application analysis)** Our research on AI app update status is divided into three aspects: **1) The update frequency of AI apps** Analyzing the update frequency of AI apps reveals how often these apps are maintained and improved. Regular updates can indicate a positive approach to enhancing app performance. **2) The correlation between AI app updates and AI model updates** Investigating the correlation between AI app updates and AI model updates helps to understand the efficiency of developers in adopting new AI technologies. **3) The correlation between AI app updates and AI framework updates.** Investigating the correlation between AI app updates and AI framework updates helps identify the dependency of apps on the latest frameworks. Understanding this relationship can guide developers in choosing suitable frameworks.
- ③ **AI framework usage analysis (Framework and model analysis)** The research on AI frameworks is mainly divided into three aspects: **1) The popularity of different AI frameworks** Understanding which AI frameworks are popular can help developers choose suitable framework among the many available options. **2) The usage**

**of single-framework and multi-framework AI frameworks** Analyzing the usage patterns of single-framework and multi-framework AI systems can help to understand different development strategies. This analysis can guide developers in choosing the most effective approach. 3) **The mainstream AI frameworks** Understanding the popularity of mainstream AI frameworks in recent years can provide insights into the popularity trends of AI frameworks, specifically which AI frameworks are gradually being phased out and which ones are on the rise. Developers can refer to the popularity of these frameworks when selecting AI frameworks.

- ④ **AI model protection analysis (Framework and model analysis)** The analysis of AI model protection is conducted from two aspects: 1) **Open-source model usage conditions** Open-source models can potentially pose higher security risks. Investigating the use of open-source models in AI apps can shed light on the state of AI model protection. 2) **AI model encryption conditions** Examining model encryption conditions is crucial for understanding to what extent on-device AI models are protected from unauthorized access.
- ⑤ **User privacy protection analysis (User analysis)** We investigate the state of user privacy protection in published AI apps. Protecting user privacy in AI apps is crucial to maintaining user trust. Analyzing the state of user privacy protection can help identify potential security vulnerabilities where improvements are needed.
- ⑥ **User review analysis (User analysis)** We investigate users' attitudes toward AI techniques in AI apps. Understanding users' attitudes toward AI techniques can help developers identify current issues that users perceive in AI apps and create AI apps that better meet user expectations and preferences.

Based on the experimental results of the above empirical analysis, we obtained a total of 23 key findings, which can be found in Section 4.3 to Section 6.2. Among these findings, Findings 1 to 10 belong to application analysis (cf. Section 4.3 and Section 4.4). Findings 11 to 18 belong to framework and model analysis (cf. Section 5.1 and Section 5.2). Finding 19 to 23 belong to user analysis (cf. Section 6.1 and Section 6.2).

In summary, our contributions are as follows:

- **Collection of a large-Scale AI App Dataset for Research** We design AI Discriminator, an automated AI application identification tool, which successfully identified 56,682 AI apps out of 7,259,232 mobile apps from the AndroZoo large-scale repository. We provide this AI app database for further research on AI apps.
- **Application Analysis** We conduct an empirical analysis of the collected 56,682 AI apps from the perspective of application popularity and update status, providing insights into the prevalence of AI apps across different

markets and categories, as well as their update frequency. This can help understand trends in AI app development and maintenance practices, guiding future AI investment and development strategies.

- **Framework and Model Analysis** We conduct an empirical analysis on the collected AI apps from the perspective of frameworks and models. We analyzed the usage of different AI frameworks, including single-framework and multi-framework approaches, and examined the state of AI model protection. This can provide insights to developers in selecting effective AI frameworks, as well as revealing potential issues in current model protection.
- **User Analysis** We conducted analysis on the collected AI apps from the perspective of users. We investigated the state of user privacy protection in AI apps and analyzed user reviews to understand user attitudes toward AI techniques applied in AI apps. This can provide valuable insights for developers to understand the current issues perceived by users and to create AI apps that better meet user expectations and preferences.

The remainder of the paper is organized as follows. Section 2 introduces the background knowledge of AI models and frameworks as well as deploying mobile ML/DL. Section 3 presents an overview of our empirical study. Section 4 shows the objective, experimental design, experimental results, and findings of our Application Analysis on AI apps. Additionally, this section also demonstrates the specific operation process of our designed AI app identification tool, AI Discriminator. Section 5 presents the Framework and Model Analysis. Section 6 presents the User Analysis. Section 7 discusses the challenges and opportunities of deploying AI technologies to mobile applications. Section 8 presents the related existing work and research. Section 9 concludes this paper.

## 2. Background

### 2.1. AI models and frameworks

Artificial Intelligence (AI) (Gamble (2020); Li, Hua, Wang, Chen and Liu (2021); Li, Dang, Ma, Klein, Traon and Bissyandé (2023); Dang, Li, Papadakis, Klein, Bissyandé and Le Traon (2023)) has become a cornerstone in the development of mobile applications, enabling enhanced functionality, user experience, and efficiency. AI models, including machine learning (ML) (Sun et al. (2021); Dang, Li, Papadakis, Klein, Bissyandé and Le Traon (2024b)) and deep learning (DL) (Huang, Hu and Chen (2021); Li, Dang, Pian, Habib, Klein and Bissyandé (2024); Dang, Li, Ma, Guo, Hu, Papadakis, Cordy and Traon (2024a)), are the foundational technologies driving the operation of apps. These models allow mobile applications to perform complex tasks such as image recognition (Jones (2020)), speech processing (Zhao, Wu and Chen (2017)), and face recognition (Amos, Ludwiczuk, Satyanarayanan et al. (2016)) with high accuracy and speed.

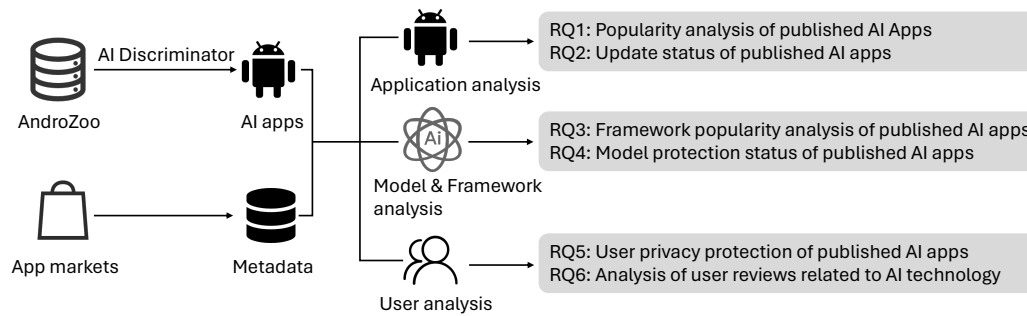


Figure 1: Overview of our explorative study.

The development and deployment of AI-driven mobile apps heavily rely on robust AI frameworks that facilitate the integration and implementation of AI techniques. Prevalent AI frameworks such as TensorFlow (Abadi, Barham, Chen, Chen, Davis, Dean, Devin, Ghemawat, Irving, Isard et al. (2016)), PyTorch (Paszke, Gross, Massa, Lerer, Bradbury, Chanan, Killeen, Lin, Gimelshein, Antiga et al. (2019)), and Keras (keras (2023)) provide developers with the tools necessary to build, train, and optimize AI models. These frameworks offer pre-built modules, extensive libraries, and flexible APIs that simplify the complex processes involved in AI development, allowing researchers to concentrate on algorithmic design rather than low-level programming, thus reducing time and labor costs.

In the context of mobile applications, specialized frameworks like TensorFlow Lite (David, Duke, Jain, Janapa Reddi, Jeffries, Li, Kreeger, Nappier, Natraj, Wang et al. (2021)) and Core ML (Thakkar and Thakkar (2019)) are designed to optimize AI models for mobile environments. These frameworks ensure that AI models can run efficiently on the limited computational resources available on mobile devices. TensorFlow Lite, for instance, leverages techniques like model quantization and hardware acceleration to enable resource-efficient execution of AI algorithms on mobile devices. Similarly, Core ML is tailored specifically for iOS devices, harnessing the power of Apple’s hardware and software integration to provide high-performance and low-latency inference for AI-driven tasks.

Moreover, in recent years, some companies introduced pre-trained models accompanied by accessible APIs, facilitating remote access to AI services (e.g., Google AI (AI (2023d)), Baidu NLP (NLP (2023)), and Amazon AI (AI (2023b))). Consequently, developers can utilize mobile AI services without necessitating expertise in AI frameworks and model design. These AI services provide ready-to-use functionalities such as image analysis and natural language processing, allowing developers to integrate advanced AI features into their apps with minimal effort.

Despite the rapid development of AI techniques, there is a growing concern regarding its environmental impact (Ali (2023)). The energy consumption and carbon emissions associated with AI data centers can be substantial, posing a significant challenge to sustainability efforts. In response to

these concerns, the concept of Green AI has emerged, which emphasizes the integration of economic and environmental sustainability into AI systems (Ali (2023)). Green AI aims to leverage machine learning to accelerate the transition to a circular economy, where products and materials are reused and recycled efficiently. The potential applications of Green AI include intelligent production planning, predictive maintenance and reuse marketplaces (Kindylidi and Cabral (2021)).

## 2.2. Deploying mobile ML/DL

Towards enabling deep learning (DL) on mobile devices, model inference can be broadly classified into two categories: cloud-based inference and on-device inference (Chen, Yao, Lou, Cao, Liu, Wang and Liu (2021)). In cloud-based inference, mobile devices connect to cloud servers over the network and send the data to the servers. ML/DL models running on the servers perform inference on the data and return the results to the mobile devices. The cloud-based approach can leverage the powerful computational resources of cloud servers to accelerate the inference process. However, cloud-based inference also has some disadvantages: **1) Privacy Risks:** Data needs to be transmitted to the cloud, which can pose privacy leakage risks. **2) Network Dependence:** It requires a stable network connection. Network latency and bandwidth limitations can affect the inference speed. **3) Cost:** Using cloud services can incur additional outsourcing costs.

On the other hand, on-device inference allows for the execution of ML/DL models directly on the mobile device, eliminating the need to send data to remote servers. On-device inference offers several advantages: **1) Privacy Protection:** Since data is processed locally on the device, it mitigates the risk of data leakage, enhancing user privacy. **2) Reduced Latency:** Without the need to transmit data over networks, on-device inference can significantly reduce latency, providing faster response times, which are crucial for real-time applications. **3) Independence from Network Connectivity:** On-device inference does not rely on internet connectivity; thus, it is not limited by network conditions and bandwidth. **4) Cost Efficiency:** On-device inference can be more cost effective as it avoids the recurring costs associated with data transmission and cloud computing services.

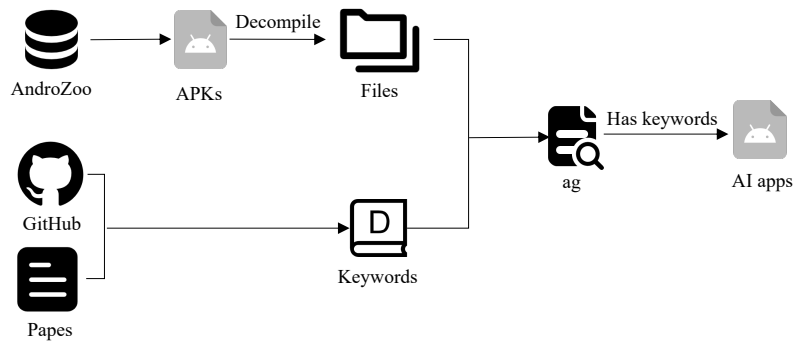


Figure 2: Pipeline of AI app Identification.

Despite these benefits, on-device inference also poses its own challenges. These include limited computational resources compared to cloud servers, which can restrict the complexity of the models that can be deployed. Additionally, the energy consumption required for local processing can be high, impacting the device’s battery life.

In the literature, several studies have explored deploying deep learning on mobile devices. Xu et al. (2019) investigated how smartphone applications utilize deep learning techniques through the analysis of over 16,500 popular Android apps, aiming to identify and characterize apps that integrate deep learning, understand their purposes, and scrutinize the deep learning models they employ. This research offers insights into current practices and potential optimization areas in mobile deep learning applications.

### 3. Analysis Overview

This paper aims to analyze AI-based applications from three perspectives: application analysis (cf. Section ??), model and framework analysis (cf. Section 5), and user analysis (cf. Section 6). To this end, we design an automated AI app extraction tool, called AI Discriminator, to recognize AI apps from the AndroZoo application database (Allix et al. (2016)), successfully collecting 56,682 AI apps from 7,259,232 Android apps.

The workflow of our study is presented in Figure 1. In the initial step, we extract AI apps from Androo using the AI Discriminator. Then, we build a crawler tool to collect important information about these AI apps from the application market. This information includes category, installs, user reviews, etc., for subsequent analysis. Specifically, our analysis focuses on the following three aspects:

- **Application analysis:** We study the extracted AI apps from the application level, including an analysis of the popularity of AI apps (RQ1) and an analysis of the update condition of AI apps (RQ2).
- **Model and framework analysis:** We analyze the internal techniques of AI apps, including the usage of AI frameworks (RQ3) and the model protection conditions of AI apps (RQ4).

- **User analysis:** We analyze AI apps from the user perspective, including privacy protection for users of AI apps (RQ5) and an analysis of user reviews (RQ6).

We discuss more details of the AI Discriminator in Section 4.1.

## 4. Applications Analysis

### 4.1. Methodology: finding AI apps

We propose **AI Discriminator** to automatically extract AI apps from the AndroZoo application repository (Allix et al. (2016)). In the following, we provide a detailed description of how AI Discriminator works and its accuracy in identifying AI apps (including precision, recall, etc.).

#### 4.1.1. Workflow of AI Discriminator

The input for the AI Discriminator is the SHA-256 hashes (256-bit Secure Hash Algorithm) of apps in AndroZoo. The output is the SHA-256 hashes of the AI apps (apps supported by AI techniques). Here, SHA-256 (256-bit Secure Hash Algorithm) is a hash function used for application encryption. AndroZoo employs it to record a unique ID for each collected app. The overall pipeline of AI Discriminator can be divided into three main steps, as illustrated in Figure 2:

- 1 **Decompilation of APK** In the first step, the AI Discriminator retrieves the APK associated with a given app. It then uses APKTool (APKTool (2023)) to unzip the package and collect its internal files.
- 2 **Building an AI Keyword Dictionary** AI Discriminator constructs an AI keyword dictionary that includes on-device ML, on-device DL, and AI service keywords. These keywords are collected based on the literature and open-source GitHub codes. Some examples of these keywords are presented in Table 1. In the following, we describe in detail how we constructed the AI keyword dictionary. Specifically, the construction of the dictionary follows the four steps below:
  - **Collecting AI Keywords from Academic Papers.** In the first step, we collected AI keywords from the existing research ( Xu et al. (2019); Sun et al. (2021)) and

their open-source repositories. These keywords have been proven effective for identifying DL/ML-based apps.

- **Collecting AI Keywords from Source Code.** Next, we analyzed the source code of each AI framework and manually collected AI-related keywords. These keywords encompass both algorithm package keywords and model-saving format keywords. To collect the model-saving format keyword, we carefully reviewed the algorithm documentation of each AI framework to identify relevant keywords. For instance, the TensorFlow Lite framework (David et al. (2021)) generally saves models in the ".tflite" format. As a result, we added ".tflite" as a keyword in our AI keyword dictionary.
- **Excluding non-AI Keywords.** Since some non-AI terms can also contain simple AI keywords (e.g., LSTM and CNN), the presence of these keywords can reduce the accuracy of the AI Discriminator in identifying AI apps. Therefore, we excluded such terms from the keyword dictionary.
- **Validation and Optimization.** We first evaluate the collected AI keywords on a small dataset. We applied all the collected AI keywords to filter out AI apps and manually checked whether they were truly AI apps. If we found any apps that were incorrectly filtered as AI apps, we identified the keywords that led to the misclassification and removed these non-AI terms from the AI keyword dictionary to optimize AI Discriminator's accuracy.

⑥ **Keyword Matching** In this step, AI Discriminator performs keyword matching using the code search tool Ag (Searcher (2023)), which is a fast and efficient code search tool designed specifically for searching large codebases. For the file packages extracted from all apps, we apply the Ag tool to scan for AI-related keywords that we collected from the previous steps. If any file in an app contains one of the keywords from the pre-defined dictionary, this app is identified as an AI application, and its SHA-256 hash and keyword information will be recorded. For example, if the file of an app contains keywords like "org.tensorflow.lite", "libtensorflowlite.so", and "N5EigenForTF Lite", this app is considered using the TensorFlow Lite framework. Additionally, in terms of the OpenCV framework, since not all OpenCV packages use AI algorithms, with some packages mainly used for image processing, we utilize the keyword "org.opencv.ml" to identify the use of AI algorithms in OpenCV. This keyword is specifically used to identify apps supported by AI-related packages in OpenCV.

#### 4.1.2. Evaluation of AI Discriminator

To validate the accuracy of the AI Discriminator in identifying AI apps, in other words, to verify whether the AI keywords in the Discriminator's keyword dictionary are

truly effective, we used five widely adopted statistical metrics for evaluation: False Positive Rate, False Negative Rate, Precision, Recall, and Accuracy. Among these, 1) False Positive Rate reflects the proportion of all non-AI apps that were incorrectly identified as AI apps. 2) False Negative Rate reflects the proportion of all AI apps that were incorrectly identified as non-AI apps. 3) Precision measures the proportion of true AI apps among all the apps identified as AI apps. 4) Recall measures the proportion of true AI apps that were correctly identified. 5) Accuracy reflects the overall correctness of the AI Discriminator in identifying AI apps.

The evaluation methodology is derived from the existing work (Sun et al. (2021)). Specifically, we manually collected and verified 450 apps, including 225 non-AI apps and 225 AI apps, as samples to evaluate the AI Discriminator. These apps were collected from existing literature sources (Xu et al. (2019); Sun et al. (2021)). The rationale behind employing a sampling approach for evaluation is that determining whether apps belong to the AI or non-AI category, namely establishing the ground truth of each app, requires manual labeling. Manually labeling all the apps can result in substantial time and labor costs. To tackle this challenge, we draw inspiration from the evaluation methodology employed in a previous study conducted by Sun et al. (2021), wherein they evaluated their developed app identification tool by sampling 438 apps. In line with this approach, we employ a similar sampling methodology for assessing the AI Discriminator based on 450 apps.

We conducted the evaluation on these 450 labeled applications. Through calculations, we obtained that the AI Discriminator's precision is 100%, the recall is 0.84, and the accuracy is 0.92. The specific calculations are as follows.

Note that in the calculations below, True Positives (TP) refers to the number of correctly predicted positive instances. True Negatives (TN) refers to the number of correctly predicted negative instances. False Positives (FP) refers to the number of instances that were incorrectly predicted as positive when they were actually negative. False Negatives (FN) refers to the number of instances that were incorrectly predicted as negative when they were actually positive. '#apps' refers to the total number of sampled apps.

- **False Positive Rate** AI Discriminator detected a total of 188 AI apps, and 0 app belong to non-AI apps. Therefore, the False Positive Rate is:

$$FPR = \frac{FP}{FP + TN} = \frac{0}{0 + 225} = 0 \quad (1)$$

- **False Negative Rate** Out of the 225 AI apps, 37 were not identified by AI Discriminator. Therefore, the FNR is:

$$FNR = \frac{FN}{TP + FN} = \frac{37}{188 + 37} = 0.16 \quad (2)$$

- **Precision** AI Discriminator detected a total of 188 AI apps, and all of these 188 apps indeed belonged to the AI

**Table 1**  
AI Frameworks given AI Keyword Dictionary

AI Frameworks Categories	AI Frameworks	Platform	Model Format
DL Frameworks	Tensorflow (Abadi et al. (2016))	Andorid, iOS	.pb, .pbtxt
	PyTorch (Paszke et al. (2019))	Andorid, iOS	.pt, .ptl
	MxNet (Paszke et al. (2019))	Andorid, iOS	.params
	Caffe (Caffe (2023))	Andorid, iOS	.caffemodel, .prototxt
	Caffe2 (Caffe2 (2023))	Andorid, iOS	.pb
	Chainer (Chainer (2023))	/	.chainermodel
	DeepLearning4j (DeepLearning4J (2023))	Andorid	.zip
	CNTK (CNTK (2023))	/	.cntk, .model
	Neuroph (Framework (2023))	/	.model, .nnet
	Lite Deep Learning Frameworks	TF Lite (David et al. (2021))	Andorid, iOS
NCNN (NCNN (2023))		Andorid, iOS	.params
Paddle Lite (Lite (2023))		Andorid, iOS	.jar, .so, etc
MACE (MACE (2023))		Andorid	.pb, .yaml
FeatherCNN (FeatherCNN (2023))		Andorid, iOS	.feathermodel
SNPE (SNPE (2023))		Andorid	/
CNNdroid (CNNdroid (2023))		Andorid	.model
TVM (TVM (2023))		/	libtvm_runtime.so, etc
Computer Vision Frameworks	OpenCV (OpenCV (2023))	Andorid, iOS	TesnorFlow, Caffe, etc
	Baidu OCR (OCR (2023))	/	/
ML Frameworks	Xgboost-predictor (ML)	/	/
	Sklearn-porter (ML, Java) (Morawiec (2021))	Andorid	/
	WEKA (ML, Java) (WEKA (2023))	/	.model
	Shogun (ML, C++) (Shogun (2023))	/	model.prof
	MALLET (ML, Java) (MALLET (2023))	/	.mallet, .classifier, etc
	Rapid Miner (ML Java) (Miner (2023))	/	.model
	Datumbbox (ML Java) (Datumbbox (2023))	/	.model
	MLPACK (MLPACK (2023))	/	/
AI Service	Google AI (AI (2023d))	/	/
	Amazon AI (AI (2023b))	/	/
	Alexa AI (AI (2023a))	/	/
	Azure AI (AI (2023c))	/	/
	Natural Language Processing Frameworks	Baidu NLP (NLP (2023))	/
Baidu Synthesizer (synthesizer (2023))		/	/

app category. Therefore, the precision of AI Discriminator is:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{188}{188 + 0} = 100\% \quad (3)$$

- **Recall** Out of the 225 AI apps, 188 were successfully identified. Therefore, the recall of AI Discriminator is:

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{188}{188 + 37} = 0.84 \quad (4)$$

- **Accuracy** Among the 450 apps, 413 apps were correctly classified. Therefore, the accuracy of AI Discriminator is:

$$\text{Accuracy} = \frac{TP + TN}{\#apps} = \frac{188 + 225}{450} = 0.92 \quad (5)$$

The evaluation above shows that out of the 225 AI apps, 37 went undetected. The main reason is that some specific keywords were not added to the keyword dictionary of the AI Discriminator to ensure that all apps identified as AI apps are indeed AI apps. For instance, '.prototxt' files are typically associated with deep learning and machine learning frameworks. However, files with the '.prototxt' format can also serve other purposes, not just AI technology, such as configuration or parameter files for software. To prevent the AI Discriminator from falsely predicting non-AI apps as AI apps, we filtered out such keywords. Consequently, this resulted in some specific AI apps being undetected.

## 4.2. Experimental Environment

We ran experiments on a high-performance computer cluster. Each cluster node runs a 2.6 GHz Intel Xeon Gold 6132 CPU with an NVIDIA Tesla V100 16G SXM2 GPU. For the data visualization, we conducted corresponding experiments on a MacBook Pro laptop with Mac OS Big Sur 11.6, Intel Core i9 CPU, and 64 GB RAM.

## 4.3. RQ1: Popularity analysis of published AI Apps

### 4.3.1. Objectives

By analyzing the popularity of AI apps, such as identifying which categories of AI apps are more popular, developers can better understand market demands, thereby improving product development and deciding in which areas to invest. In this research question, we analyzed the popularity of AI apps through 4 sub-questions:

- **RQ-1.1** Has the released ratio of AI apps increased rapidly in recent years?
- **RQ-1.2** In which markets are AI apps mainly released?
- **RQ-1.3** Which AI app categories do the top providers prefer?
- **RQ-1.4** Which categories of AI apps are more prevalent in the market?

### 4.3.2. Experimental Methodology

First, we utilize AI Discriminator (cf. Section 4.1) to extract AI apps from the AndroZoo application database (Allix et al. (2016)). Next, we analyze these AI apps using the following methodology.

- **Experiment RQ-1.1 (AI app released ratio)** To calculate the proportion of AI apps among all apps published each year, we record the total number of apps published and the number of AI apps published each year, and use Formula 6 for calculation:

$$\text{Ratio} = \frac{\# \text{ AI apps}^y}{\# \text{ published apps}^y} \quad (6)$$

where  $y$  refers to the specific year.  $\# \text{AI apps}$  refers to the number AI apps published in the year  $y$ .  $\# \text{published apps}$  refers to the number of all the apps published in the year  $y$ .

- **Experiment RQ-1.2 (AI app market distribution)** To investigate the distribution of AI apps across different markets, we first obtained information from AndroZoo about which market each AI app belongs to. Then, we calculated the number of AI apps released by each market.
- **Experiment RQ-1.3 (App provider’s preference for AI apps)** To investigate which categories AI apps providers are more focused on, we crawl the provider information for all the collected AI apps from the application market. The crawled information includes the app provider each app belongs to, categories, number of downloads, ratings, etc. We then calculate the number of AI apps developed by each app provider and identify the top ten providers based on the number of AI apps developed. Additionally, we analyze and report the category information of AI apps developed by each of these top ten providers.
- **Experiment RQ-1.4 (Market’s preference for AI apps)** To investigate the popularity of different AI app categories in the market, we use a crawler tool to collect metadata for each app, including its category, number of installs, and rating. We then calculate the total number of apps, average number of installs, and average rating for each category.

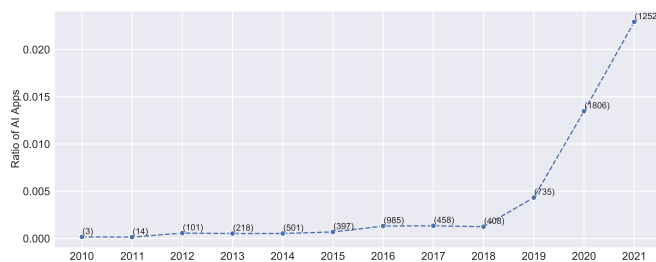


Figure 3: Ratio of AI apps developed in each year.

### 4.3.3. Results

Figure 3 presents the experimental results of **RQ-1.1**, showing the proportion of AI apps among the apps released each year. The X-axis exhibits the years, and the Y-axis presents the percentage of AI apps. From Figure 3, we see that from 2010 to 2018, the proportion of AI app development grew relatively slowly, while it rose sharply from 2018 to 2021. The experimental results indicate that from 2018, incorporating AI technology into applications has become a growing trend in mobile app development.

**Finding 1:** Since 2018, incorporating AI technology into applications has become a growing trend in mobile app development.

Figure 4 presents the experimental results for **RQ-1.2**, showcasing the market distributions for AI app releases. The X-axis displays the names of the markets, and the Y-axis indicates the number of AI apps released by each market. We see that Google Play has the highest number of AI apps published, with a total of 50,268 AI applications, which is 12 times more than the second highest, Anzhi, which released 4,079 AI apps. Following Anzhi are AppChina, VirusSharing, and PlayingDrones, which released 1,544, 849, and 645 AI apps, respectively.

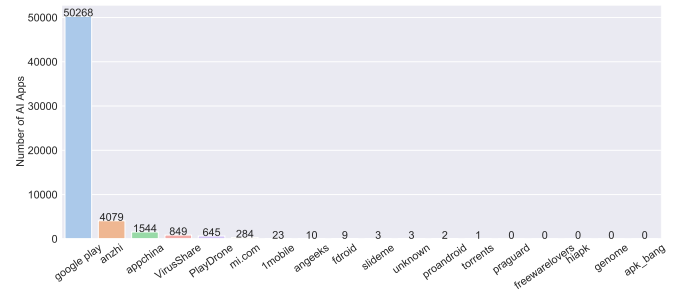


Figure 4: Number of AI apps in each market.

**Finding 2:** Currently, the market with the highest number of AI app releases is Google Play, significantly surpassing other markets. Other markets that have released over 1000 AI apps include Anzhi and AppChina.

Table 2 presents the experimental results of **RQ-1.3**, showing the main categories of AI apps developed by the top-10 AI app providers. From left to right, the columns are the provider name, the number of AI apps each provider developed, and the main categories of these apps. We see that the top-10 AI app providers developed more AI apps in the Food & Drink, Business, and Education categories. Specifically, 60% of the companies published AI apps in Food & Drink, 50% released Business-oriented AI apps, and 40% published Education-oriented AI apps.

**Finding 3:** Top-10 AI app providers developed more AI apps in the Food & Drink, Business, and Education categories.



**Table 2**  
Top 10 Providers of AI apps

Provider name (Top 10)	Number of AI app	Category
Delivery Direto by Kekanto	353	Food & Drink, Business, Shopping
CloudFaces	129	Food & Drink, Business, Education, Beauty, Social, Entertainment, Travel & Local, Medical, Sports, Health & Fitness
Tom McLeod Software	102	Productivity
Laboratory X	96	Food & Drink, Lifestyle
Klass Apps, Inc.	87	Education, Entertainment
Core-apps	85	Books & Reference
Tara Blooms Private Ltd	85	Food & Drink, Business, Education, Finance, Beauty, Travel & Local, Lifestyle, Shopping, Medical, Auto & Vehicles, Events
Tiffin Tom Ltd	75	Food & Drink
InnoShop Co	70	Food & Drink, Business, Education, Entertainment, Beauty, Shopping
Via Transportation Inc.	70	Business, Maps & Navigation, Travel & Local,

The experimental results of **RQ-1.4** are presented in Table 3, which shows the popularity of AI apps across different categories. In Table 3, we present the number of AI apps in each category, their proportions (the ratio of AI apps to all apps), average installations, and average scores. Based on the analysis of Table 3, we obtained the following findings (i.e., finding 4 to finding 7):

In Table 3, the column "Ratio" refers to the proportion of AI apps to all apps within each category. The data in the table is sorted in descending order based on this ratio. We see that the financial field has the highest proportion of AI apps, accounting for 7.28%. Other categories with a high proportion of AI apps are Photography, Productivity, Auto & Vehicles, Food & Drink, Libraries & Demo, and Parenting, accounting for 4.75%, 3.80%, 3.68%, 3.27%, 3.13%, and 3.11%, respectively.

**Finding 4:** The financial field has the highest proportion of AI apps, followed by Photography, Productivity, Auto & Vehicles, Food & Drink, Libraries & Demo, and Parenting.

In Table 3, the column "Count" demonstrates the number of published AI apps in each category. We see that the categories Finance and Business released the highest number of AI apps, with 2845 and 2398 apps, respectively. This is followed by Education, Productivity, and Tools, with 1920, 1752, and 1322 AI apps released, respectively. In contrast, some areas have relatively few AI apps released, including Dating, Racing, Role-Playing, and Strategy games.

**Finding 5:** The categories Finance and Business released the highest number of AI apps, followed by Education and Tools.

In Table 3, the column "Avg Installs" demonstrates the average installs of AI apps in each category. We see that AI apps in the Strategy Games category have the highest number of installs, with 5,557,578 installs on average. This is followed by the Racing and Productivity categories, with 3,500,191 and 2,969,649 installs, respectively.

**Finding 6:** AI apps in the Strategy Games category have the highest number of installs, followed by the Racing and Productivity categories.

**Table 3**  
Categories of AI apps

Category	Ratio	Count	Avg Installs	Avg Score
Finance	7.28%	2845	143337	4.1
Photography	4.75%	807	1639865	3.7
Productivity	3.80%	1752	2969649	3.8
Auto & Vehicles	3.68%	374	18443	3.6
Food & Drink	3.27%	1322	53698	4.0
Libraries & Demo	3.13%	84	7535	3.6
Parenting	3.11%	62	57605	3.6
Business	3.01%	2398	19958	3.8
Beauty	2.88%	174	35702	3.7
Shopping	2.54%	1102	264894	4.0
Medical	2.39%	426	9105	3.7
Events	2.38%	158	739	4.7
Communication	2.34%	512	2011779	3.7
House & Home	2.20%	173	12817	3.6
Tools	2.13%	1710	219064	3.6
Travel & Local	2.13%	764	38989	3.8
Art & Design	2.05%	185	20539	3.4
Social	1.83%	165	78650	3.7
Health & Fitness	1.79%	865	26313	3.6
Lifestyle	1.68%	985	50755	3.6
Maps & Navigation	1.63%	237	163105	3.7
Entertainment	1.63%	970	48776	3.5
Sports	1.58%	389	11748	3.7
Video Players & Editors	1.34%	85	2128589	3.7
Education	1.30%	1920	20266	3.8
Comics	1.28%	10	1476	4.4
Weather	0.98%	39	13449	4.0
News & Magazines	0.95%	211	8531	4.3
Books & Reference	0.52%	299	10549	4.1
Music & Audio	0.28%	161	36611	4.1
Dating	0.22%	1	50	0.0
Personalization	0.21%	79	2717519	3.8
Trivia	0.43%	27	1546	3.7
Word	0.33%	17	765	2.9
Strategy	0.22%	9	5557578	2.9
Card	0.99%	39	44559	3.8
Music	0.64%	25	82	1.0
Board	0.63%	34	8130	4.1
Casual	0.26%	72	36145	3.7
Puzzle	0.21%	61	97574	4.1
Role Playing	0.20%	7	1436601	4.2
Simulation	0.17%	19	326916	3.6
Adventure	0.16%	16	64839	4.0
Action	0.15%	16	641959	3.9
Racing	0.10%	6	3500191	4.4
Arcade	0.09%	27	1872319	3.2

In Table 3, the column "Avg Score" shows the average score of AI apps in different categories, ranging from 0 to 5. We see that the categories with the highest average scores are Racing Games and Comics, both obtaining a score of 4.4. Additionally, 91.3% of the categories have an average score above 3.0, and 87.0% of the categories have an average score above 3.5.

**Finding 7:** AI apps in the Racing Games and Comics categories have the highest average scores. 91.3% of the categories have an average score above 3.0, and 87.0% of the categories have an average score above 3.5.

#### 4.4. RQ2: Update status of published AI apps

##### 4.4.1. Objectives

We aim to investigate the updates of AI apps, including the update frequency of AI apps, the correlation between app updates and embedded model updates, and the correlation

between app updates and framework updates. Analyzing the update frequency of AI apps can reveal how often AI apps are maintained. Investigating the correlation between AI app updates and AI model updates can contribute to understanding the efficiency of developers in adopting new AI technologies. Understanding this relationship between AI app updates and AI framework updates can guide developers in selecting suitable frameworks. Specifically, we analyzed the updates of AI apps through three sub-questions:

- **RQ-2.1** How fast are the apps on the market updated?
- **RQ-2.2** To what extent do AI app updates accompany AI model updates?
- **RQ-2.3** Which AI framework transitions typically occur when AI apps are updated?

#### 4.4.2. Experimental Methodology

We conduct the following three experiments to answer the sub-questions.

- **Experiment RQ-2.1 (AI app update frequency)** To investigate the update frequency of the collected AI apps, we obtained the update information for each AI app from AndroZoo. In AndroZoo, applications under the same package name are different versions of the original app. We leveraged these package names to find out how many times each AI app has been updated.
- **Experiment RQ-2.2 (The correlation between app updates and model updates)** First, based on RQ-2.1, for each AI app, we identified its previous historical version apps. For each AI app and its historical versions, we downloaded the related application APKs from AndroZoo and decompiled them to obtain the internal files. We designed a program that can automatically extract the embedded models of each app and apply a hash function to calculate hash values for the extracted models. By comparing the hash values of the models in the original AI apps with those in their historical versions, we can determine whether the models have been updated.
- **Experiment RQ-2.3 (The correlation between app updates and framework updates.)** Similarly, based on RQ-2.1, for each AI app, we identified its previous historical version apps. Then, we fed each AI app and its historical versions into AI Discriminator to obtain their applied AI frameworks and observed the changes in AI frameworks when the AI apps were updated.

#### 4.4.3. Results

Figure 5 presents the experimental results of **RQ-2.1**, showcasing the update frequency of the collected AI apps. The  $n$  represents the number of updates. We separate all the collected AI apps into five groups based on the number of updates and present the proportion of AI apps in each category. In Figure 5, we see that the majority of AI apps (80.3%) updated no more than 5 times. 10.3% of AI apps are

updated between 5 to 10 times. Only 4.3% of AI apps are updated more than 20 times.

**Finding 8:** The majority of AI apps (80.3%) updated 0-5 times.

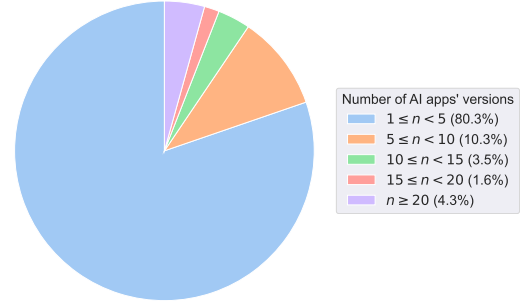


Figure 5: Distribution of historical version of AI apps.

In response to **RQ-2.2**, we performed data analysis. Out of the 23,466 AI apps collected, 4,818 AI apps were updated, of which 2,225 had their AI models updated. Therefore, based on Formula 7, the proportion of model updates among the updated AI apps is 46.18%.

$$\text{Ratio} = \frac{\#model\ updates}{\#AI\ app\ updates} = \frac{2225}{4818} = 46.18\% \quad (7)$$

where #model updates refer to the number of updated AI apps whose models were also updated. #AI app updates refer to the number of updated AI apps.

**Finding 9:** Among all the updated AI apps, the proportion of AI apps whose models were also updated is 46.18%.

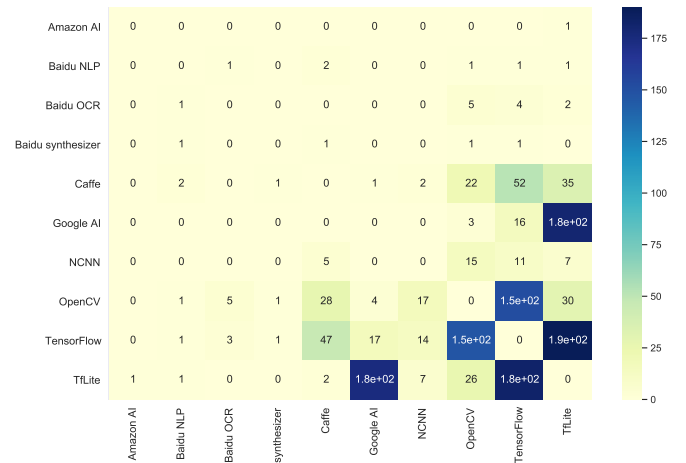


Figure 6: AI Frameworks change when AI apps are updated.

The experimental results of **RQ-2.3** are presented in Figure 6, which demonstrates the updates of AI frameworks when AI apps are updated. The X-axis refers to the AI framework after updates, and the Y-axis represents the frameworks before updates. The numbers in the grid indicate the

number of transitions from the Y-axis framework to the X-axis framework, with darker colors indicating more frequent transitions. In Figure 6, we see that the top two most frequent framework transitions are TensorFlow → TFLite and TFLite → TensorFlow. Additionally, two other frequently occurring transitions are Google AI → TFLite and TFLite → Google AI.

**Finding 10:** The most frequent framework transitions are TensorFlow → TFLite, TFLite → TensorFlow, Google AI → TFLite and TFLite → Google AI.

## 5. Framework and Model Analysis

### 5.1. RQ3: Framework popularity analysis of published AI apps

#### 5.1.1. Objectives

AI frameworks are utilized to build and deploy AI models (Abadi et al. (2016)). Some popular AI frameworks include TensorFlow, PyTorch, etc. In this research question, we investigate the popularity of different AI frameworks, the prevalence of single-framework and multi-framework AI applications, and the popularity trends of mainstream AI frameworks in recent years. We present our motivation for investigating these three aspects as follows.

Understanding which AI frameworks are popular can help developers select more suitable frameworks among the many available AI frameworks. Investigating the usage condition of single-framework and multi-framework AI systems can help understand the prevalence of different development strategies. Understanding the popularity of mainstream AI frameworks can provide insights into the popularity trends of AI frameworks, specifically which AI frameworks are gradually being phased out and which ones are on the rise. Specifically, we analyzed AI frameworks based on three sub-questions:

- **RQ-3.1** How do on-device ML, on-device DL, and AI service apps differ in terms of usage and size?
- **RQ-3.2** What is the proportion and prevalence of AI apps using single and multiple AI frameworks?
- **RQ-3.3** Among the mainstream AI frameworks, which frameworks have been the most popular in recent years?

#### 5.1.2. Experimental methodology

We conducted the following three experiments to answer the sub-questions above.

- **Experiment RQ-3.1 (Analysis on on-device ML, on-device DL, and AI service apps in terms of usage and size)** Existing AI frameworks can be broadly divided into three categories: on-device ML apps, on-device DL apps, and AI service apps. On-device ML apps refer to AI apps that perform inference using classical ML models deployed on mobile devices. On-device DL apps use DL models deployed on mobile devices for inference. AI service apps use cloud-based ML/DL services for

inference. Here, we analyze these three types of AI apps from the following perspectives: application quantity and application size. 1) We calculated the number of AI apps in each category and the proportion of AI apps within each category. 2) We obtained the size information for each AI app and analyzed the size distribution of AI apps within each category. Moreover, we further categorized the AI frameworks and model format into six main categories, namely: 1) Traditional Deep Learning Frameworks, 2) Lite Deep Learning Frameworks, 3) Classical Machine Learning Frameworks, 4) Computer Vision Frameworks, 5) General Cloud AI Frameworks, and 6) Natural Language Processing Frameworks. In Table 1, we present the category to which each framework/model format belongs. In the following, we introduce each category and provide corresponding examples.

- **Traditional Deep Learning Frameworks** are typically used for training and deploying deep neural networks. Typical examples include TensorFlow and PyTorch.
- **Lite Deep Learning Frameworks** are mainly used for deploying deep learning models on mobile devices. They are typically lightweight and can run efficiently on hardware with limited computational power. Typical examples include TensorFlow Lite and NCNN.
- **Classical Machine Learning Frameworks** are mainly used for implementing traditional machine learning algorithms, such as decision trees, Support Vector Machines (SVMs), k-nearest Neighbors (KNN), etc. Typical frameworks include Sklearn-porter and WEKA.
- **Computer Vision Frameworks** are specifically designed for image processing and computer vision tasks. OpenCV is a commonly used computer vision framework.
- **General Cloud AI Frameworks** are provided by cloud service providers and are designed to leverage the powerful computational capabilities and large-scale data processing capabilities of cloud computing for training and deploying AI models. Typical examples include Google AI Platform, Amazon AI, and Microsoft Azure AI.
- **Natural Language Processing Frameworks** focus on processing and analyzing natural language data, including tasks like semantic analysis. Typical examples include Baidu NLP.

- **Experiment RQ-3.2 (Single-framework and multi-framework AI apps)** We investigate the usage of single and multiple frameworks in AI apps. Single-framework AI apps refer to AI apps that use only one AI framework, while multi-framework AI apps use two or more frameworks. In RQ2.3 (cf. Section 4.4), we have obtained the framework information for each AI app. Based on this information, we ranked the frameworks/framework combinations according to their popularity and identified

the top 10 most prevalent frameworks/framework combinations.

- **Experiment RQ-3.3 (Mainstream AI frameworks)** We investigated the popularity of 16 mainstream frameworks in recent years. We counted the usage frequency of the mainstream AI frameworks for each time period.

### 5.1.3. Results

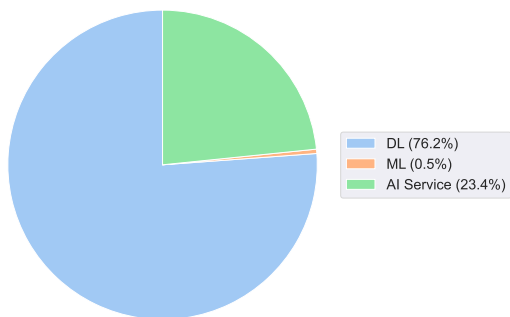


Figure 7: Ratio of AI apps in DL, ML and AI Service.

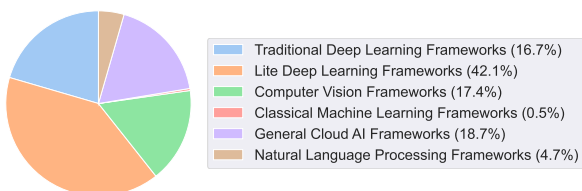


Figure 8: Ratio of AI apps across different framework categories.

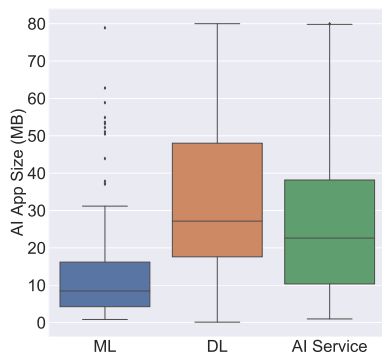


Figure 9: AI app size in DL, ML and AI Service

The experimental results of **RQ-3.1** are presented in Figure 7, Figure 8, and Figure 9. Figure 7 presents the proportions of the three types of AI apps: on-device ML, on-device DL, and AI service. From Figure 9, we see that on-device DL apps have the highest proportion (76.2%). Next are AI service-based apps, accounting for 23.4%. Finally, on-device ML apps have the lowest proportion, accounting for only 0.5%.

Figure 8 presents a more detailed breakdown of the collected AI apps. As shown in the figure, Lite Deep Learning Frameworks have the highest proportion, accounting for 42.1%. General Cloud AI Frameworks rank second, accounting for 18.7% of the collected AI apps. Next are Computer Vision Frameworks and Traditional Deep Learning Frameworks, which account for 17.4% and 16.7%, respectively. To conclude, Lite Deep Learning Frameworks, General Cloud AI Frameworks, Computer Vision Frameworks, and Traditional Deep Learning Frameworks are the frameworks with the highest proportions among the collected AI apps.

**Finding 11:** AI apps supported by on-device DL techniques accounted for the highest proportion. Lite Deep Learning Frameworks, General Cloud AI Frameworks, Computer Vision Frameworks, and Traditional Deep Learning Frameworks are the frameworks with the highest proportions among the collected AI apps.

Figure 9 shows the size distribution of three different categories of AI apps. We see that on-device DL apps have the highest median, approximately 27 MB. The size distribution mainly ranges from 20 MB to 48 MB. AI service apps have the second-highest median, around 23 MB, with a main range of 10 MB~40 MB. On-device ML apps have the lowest median size, around 8 MB, with a main size range of approximately 5 MB~16 MB.

**Finding 12:** On-device DL apps have a relatively larger size compared to AI service apps and on-device ML apps.

The experimental results of **RQ-3.2** are presented in Figure 10, which shows the usage of single framework and multiple frameworks in the collected AI apps.

Figure 10(a) shows the proportion of collected AI apps using one AI framework, two AI frameworks, and three AI frameworks. We can see that AI apps using one AI framework account for the largest proportion (75.2%). AI apps using two frameworks account for the second largest proportion, at 21.3%. AI apps using three frameworks account for the smallest proportion, at 3.3%.

**Finding 13:** Among all the collected AI apps, single-framework AI apps account for the highest proportion.

Figure 10(b) exhibits which AI frameworks are more popular among AI apps using a single framework. We can see that AI apps using the AI framework TFLite account for the highest proportion, at 41.5%. This is followed by OpenCV, TensorFlow, and Google AI, making up 16.5%, 15.8%, and 12.9%, respectively.

**Finding 14:** Among AI apps using a single framework, apps using the TFLite framework have the highest proportion, at 41.5%.

Figures 10(c) and Figure 10(d) show the prevalence of different AI framework combinations. Specifically, Figure 10(c) displays the most popular two-framework combinations, while Figure 10(d) shows the most popular three-framework combinations. In Figure 10(c), we see that the

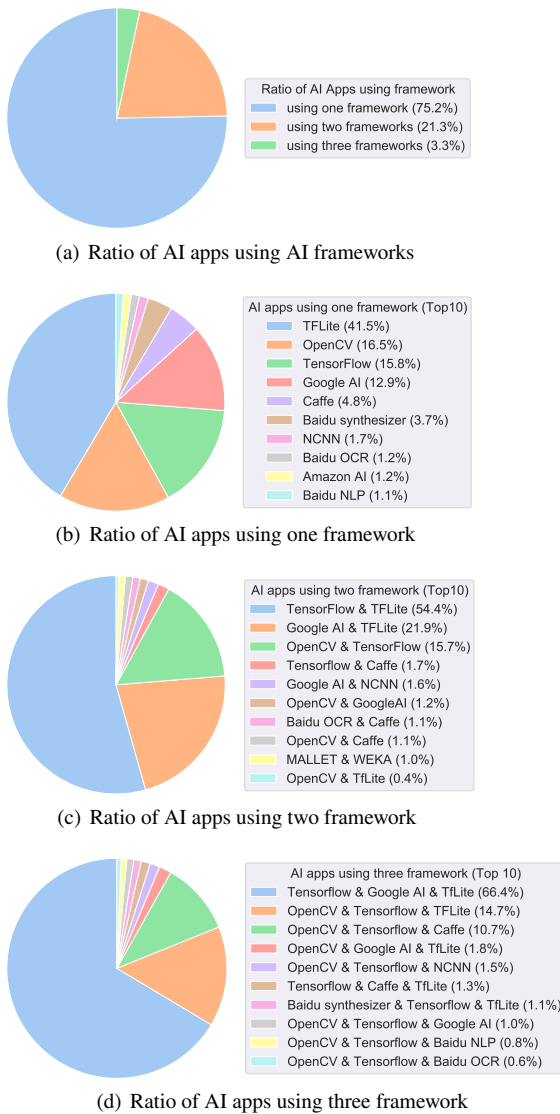


Figure 10: Distribution of AI apps using AI frameworks.

combination of TensorFlow & TFLite is the most popular. About 54.4% of the collected two-framework AI apps use this combination. This is followed by Google AI & TFLite and OpenCV & TensorFlow, accounting for 21.9% and 15.7%, respectively. Regarding AI apps using three frameworks, the combination of TensorFlow & Google AI & TFLite is the most popular, accounting for 66.4%. This is followed by OpenCV & TensorFlow & TFLite and OpenCV & TensorFlow & Caffe, accounting for 14.7% and 10.7%, respectively.

**Finding 15:** Among AI apps using two frameworks, apps using the combination of TensorFlow & TFLite are the most popular, accounting for 41.5%. Among AI apps using three frameworks, apps using the combination of TensorFlow & Google AI & TFLite are the most popular, accounting for 66.4%.

The experimental results of **RQ-3.3** are demonstrated in Figure 11, which presents the popularity of 16 different AI frameworks from 2011 to 2021. The X-axis represents the year, and the Y-axis represents the AI frameworks. In Figure 11, the size of the points indicates the usage frequency of AI frameworks applied in each time slice by the newly released AI apps. The larger the point, the more frequently the framework was used in that year. From Figure 11, we can see that OpenCV has the largest point areas, indicating it has been consistently popular. On the other hand, TFLite and Google AI have rapidly become prevalent since 2020. Additionally, TensorFlow was consistently used between 2018 and 2021. The experimental results show that from the perspective of usage frequency, OpenCV, TFLite, Google AI, and TensorFlow are relatively popular compared to other AI frameworks.

**Finding 16:** From the perspective of usage frequency, some relatively more popular AI frameworks are OpenCV, TFLite, Google AI, and TensorFlow.

## 5.2. RQ4: Model protection status of published AI apps

### 5.2.1. Objectives

Sufficient model protection can prevent the core intellectual properties of AI apps from being stolen by malicious competitors. However, the boom in on-device AI apps increases the likelihood of model leaks as models are deployed directly on the client side (Deng, Chen, Meng, Zhang, Xu and Cheng (2022)). In this research question, we investigate the model protection condition in AI apps. We conducted our study from two aspects: 1) We examined the use of open-source models in AI apps, as open-source models potentially pose higher security risks. 2) We investigated the encryption level of models embedded in AI apps, as strong encryption can significantly prevent AI app models from being stolen.

The motivation for investigating the above two issues is twofold: 1) Open-source models can potentially pose security risks. Investigating the use of open-source models in AI apps can, to some extent, shed light on the state of AI model protection. 2) Investigating model encryption conditions can help understand to what extent on-device AI models are protected. We summarize the above two considerations into the following two sub-questions.

- **RQ-4.1** To what extent do public AI apps use open-source models to perform AI tasks?
- **RQ-4.2** Are models embedded in published AI applications well-encrypted?

### 5.2.2. Experimental methodology

We conduct two corresponding experiments to answer the above two sub-questions.

- **Experiment for RQ-4.1 (Usage of open-source models in public AI apps)** We conducted experiments to calculate the ratio of AI apps directly using some common

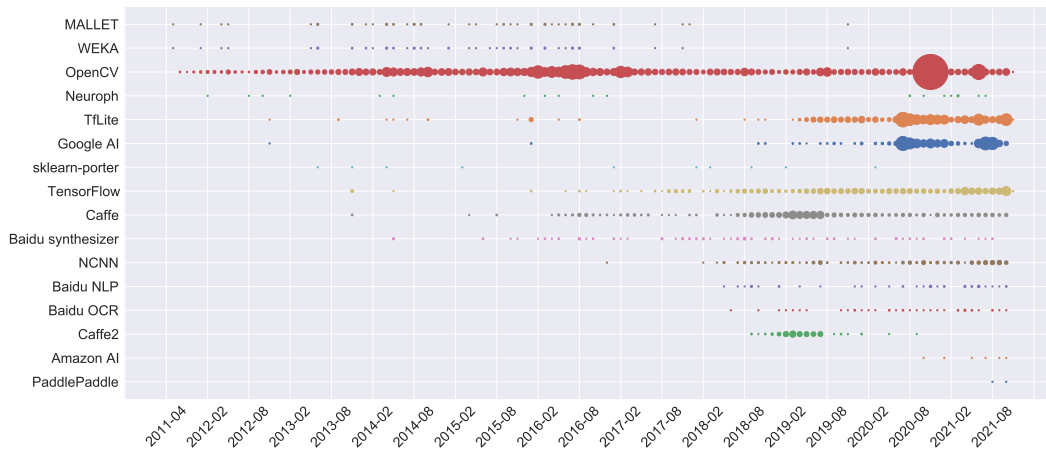


Figure 11: The usage of AI framework each year.

open-source models among the collected AI apps. In the first step, we collected 90 open-source mobile models from the TFLite Hub (Hub (2023)), a repository that provides reusable ML models and calculated their hashes. TFLite was chosen as the source for open-source models because it is one of the most widely used framework codebases for deploying machine learning models on mobile devices. In the second step, we calculated the hashes of our collected 23,466 AI app models. For each AI app model, we matched its hash with the collection of open-source model hashes. If the hash of a model in an AI app matched any of the hashes in the open-source model collection, we considered that the AI app used open-source models. Using this method, we calculated the proportion of the AI apps that utilized common open-source models.

- **Experiment for RQ-4.2 (Model encryption status of AI apps)** To study the encryption status of collected AI app models, we leverage the idea of the standard entropy test (Sun et al. (2021)) to determine whether a given AI model is encrypted. The standard entropy test assesses whether an AI model is encrypted by calculating its entropy. According to its principle, encrypted AI models typically exhibit high entropy values. Following the existing study (Sun et al. (2021)), we set the entropy threshold for encryption at 7.99. If an AI model file’s entropy exceeds this value, we consider the AI model encrypted.

Moreover, to prove the effectiveness of this approach, we conducted the following evaluation: First, we applied the standard entropy test approach with the threshold of 7.99 to obtain a set of AI models considered encrypted by this method. We randomly selected 50 models from this set and attempted to open them manually using the model viewer Netron. We found that all the models considered encrypted could not be opened. Next, we tried to use the official API of these AI models to load these models. Generally, unencrypted models can be successfully loaded. We also found that none of the models could be opened. Through

Table 4 Example of public models renamed in AI apps

Public models	Task	Renamed models in AI apps
magenta-arbitrary-image-stylization-v1-256-int8-transfer-1.tflite	Image-style-transfer	art-photo-384.tflite, transfer-model.tflite, style-transfer-quantized-384.tflite, style-transfer.tflite
lite-model-aiy-vision-classifier-insects-V1-3.tflite	Image-classification	insects-C.tflite, aiy-classifier-natural-world-insects-V1-2-quantized-input-ui-n8-85018f9a4c0110bd69f70-be107f7d2207124c301-model-with-metadata.tflite
lite-model-ssd-mobilenet-v1-1-metadata-2.tflite	Image Object Detection	detect.tflite

these methods, we concluded that the standard entropy test with a threshold of 7.99 is a reasonable and effective method for verifying whether models are encrypted.

### 5.2.3. Results

The experimental results of RQ-4.1 are presented in Table 4 and Table 5. Among the 23,466 AI app models collected, we found that 175 apps utilize open-source models by simply renaming them, accounting for 0.7%. Since malicious attackers can easily obtain the input and output shapes of the open-source models, directly using open-source models exposes AI apps to security threats. Table 4 shows some examples of directly renamed and used open-source AI models. The table from left to right presents the model name, model tasks, and company-provided renames. Table 5 presents ten prevalent published models used by providers. From left to right, the table shows the model names, model execution tasks, and providers using these models. We see that the mainstream execution tasks of these models are image classification and image object detection.

**Finding 17:** Among the collected AI apps, 175 utilized open-source AI models. The mainstream execution tasks of the ten prevalent open-source AI models are image classification and image object detection.

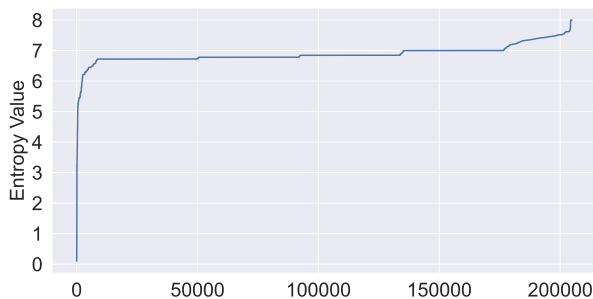
The experimental results of RQ-4.2 are presented in Figure 12, which shows the encryption status of the collected

**Table 5**  
Examples of public mobile models used in provider

public mobile models	Task	Providers
efficientnet-lite0-int8-2.tflite	Image-classification	Shopping Deals & Specials, edobo
efficientnet-lite4-int8-2.tflite	Image Classification	Dave Bennett
lite-model-aiy-vision-classifier-birds-V1-3.tflite	Image Classification	DSM Services, Vanchel
lite-model-aiy-vision-classifier-food-V1-1.tflite	Image Classification	AG Apps Co
mobilenet-v2-1.0-224-1-metadata-1.tflite	Image Classification	Memorizer
lite-model-cropnet-classifier-cassava-disease-V1-1.tflite	Image Classification	Solomon Nsumba
lite-model-object-detection-mobile-object-labeler-v1-1.tflite	Image Classification	Glitter Technology Ventures LLC
lite-model-ssd-mobilenet-v1-1-metadata-2.tflite	Image Object Detection	A La Carte Media Inc., Apptastic Mobile, Bridgewiz Engineering, DistinctView, Farid Ahmad Ahmadyar, FavLabs, LazyDroid, MKK Games, MLPJ DROID, PlatineX TDC, Polycents, RajAppStudio, SANE Tech, Sifie Apps, Sparkling India, TeamLease EdTech Ltd., Yusuf Suhair
object-detection-mobile-object-localizer-v1-1-default-1.tflite	Image Object Detection	FRUCT, Nexart TechnoSolutions Pvt Ltd
lite-model-cartoongan-int8-1.tflite	Image Style Transfer	Dan Group, Dotsquares, Pixel Force Pvt Ltd

AI app models. As mentioned in the experimental design of RQ-4.2, we used the standard entropy test (Sun et al. (2021)) to determine whether an AI model is encrypted. If an AI model’s entropy exceeds 7.99, this model is considered encrypted. In Figure 12, the Y-axis represents the entropy value of a model, while the X-axis represents the index of each AI model. All AI models are sorted by their entropy values from left to right. From Figure 12, we can see that the number of AI models with entropy values exceeding 7.99 is small. After manual verification, we found that the number of models considered to be encrypted is 520, accounting for 0.25%.

**Finding 18:** Among all the collected AI models, the number of models considered to be encrypted is 520, accounting for only 0.25%.



**Figure 12:** Model entropy of AI apps.

## 6. User Analysis

### 6.1. RQ5: User privacy protection of published AI apps

#### 6.1.1. Objectives

Protecting user privacy in AI apps is crucial to maintaining user trust. Failure to adequately protect user data can lead to security breaches, misuse of personal information, and reputational damage for providers. In this research question, we investigate the state of user privacy protection in published AI apps. Analyzing the state of user privacy protection can contribute to identifying potential privacy protection vulnerabilities where improvements are needed.

Specifically, we analyzed the state of user privacy protection in AI apps from the following perspectives:

- **RQ-5.1** To what extent is user privacy protected in AI apps?
- **RQ-5.2** What are the main privacy concerns of users regarding current AI apps?

#### 6.1.2. Experimental methodology

We conducted the following two experiments to answer the sub-questions above.

- **Experiment RQ-5.1 (Analysis on user privacy protection)** In the first step, we construct a privacy keyword list containing common user privacy terms (e.g., name, email, address, and birth). Then, we crawl the privacy policy data of 6016 AI apps from the Google Play market and match it with the pre-built privacy keyword list. If the crawl data includes keywords from the privacy list, we consider that the corresponding AI apps can access private data with respect to these keywords.
- **Experiment RQ-5.2 (User concerns on current AI app privacy protection)** First, we filtered privacy-related comments using privacy-related keywords (such as “privacy”) from all the reviews of collected AI apps. Next, we manually analyzed and summarized the collected privacy-related user comments, aiming to identify the core concerns users have about AI apps’ privacy protection. Based on the manual analysis, we identified several core concerns, which are: 1) Privacy infringement and data misuse; 2) Lack of transparency in privacy policies; 3) Third-party data sharing; 4) Privacy protection features; 5) The conflict between privacy and user experience. Finally, we summarized and analyzed each of these concerns.

#### 6.1.3. Results

Table 6 presents the experimental results for RQ5. It shows the condition of user private data being accessed in six different categories of AI apps. The column "Count" indicates how many AI apps accessed this private content, and "Ratio" represents the proportion of AI apps that accessed this content. From Table 6, we can see that the most frequently accessed private attribute across all six categories is the user’s name, indicating that this private attribute is the

**Table 6**  
Private data of AI apps accessing

Business			Finance			Education		
	Count	Ratio		Count	Ratio		Count	Ratio
privacy	492	0.60	privacy	410	0.52	privacy	333	0.61
name	421	0.51	name	314	0.39	name	292	0.53
address	405	0.50	location	311	0.39	address	250	0.45
email	363	0.44	email	305	0.38	email	201	0.36
location	335	0.41	address	303	0.38	location	195	0.35
image	302	0.37	image	287	0.36	image	138	0.25
account	211	0.26	phone number	166	0.21	account	104	0.19
country	203	0.25	credit card	114	0.14	video	96	0.17
phone number	196	0.24	country	112	0.14	credit card	88	0.16
video	151	0.18	video	110	0.13	phone number	87	0.15
interaction	141	0.17	interaction	110	0.13	device name	81	0.14
credit card	131	0.16	birth	85	0.10	country	80	0.14
photo	123	0.15	audio	71	0.09	interaction	57	0.10
birth	88	0.10	photo	48	0.06	photo	51	0.09
device name	84	0.10	gender	44	0.05	gender	40	0.07
audio	68	0.08	device name	28	0.03	audio	33	0.06
gender	28	0.03	user ID	15	0.01	birth	15	0.02
device ID	21	0.02	device ID	9	0.01	device ID	10	0.01
user ID	15	0.01	browsing history	6	0.00	user ID	6	0.01
browsing history	6	0.00	search history	4	0.00	browsing history	2	0.00
search history	2	0.00	IMEI number	2	0.00	IMEI number	2	0.00
IMEI number	1	0.00	IMEI number	1	0.00	Android ID	2	0.00
Android ID	0	0.00	frequency of use	0	0.00	frequency of use	2	0.00
frequency of use			Android ID			search history	1	0.00
Productivity			Tools			Shopping		
	Count	Ratio		Count	Ratio		Count	Ratio
privacy	371	0.72	privacy	289	0.66	privacy	173	0.63
name	328	0.64	name	241	0.55	name	155	0.56
address	324	0.63	address	225	0.52	address	151	0.55
email	167	0.32	email	185	0.42	email	142	0.51
location	154	0.30	location	185	0.42	location	122	0.44
image	147	0.28	image	152	0.35	account	118	0.43
account	109	0.21	account	104	0.24	image	81	0.29
country	91	0.17	phone number	95	0.21	country	71	0.25
phone number	72	0.14	country	90	0.20	phone number	52	0.18
interaction	69	0.13	video	90	0.20	video	47	0.17
video	56	0.11	device name	74	0.17	interaction	47	0.17
photo	52	0.10	photo	71	0.16	credit card	47	0.17
device name	47	0.09	interaction	60	0.13	gender	32	0.11
credit card	44	0.08	audio	51	0.11	photo	32	0.11
audio	25	0.04	credit card	46	0.10	birth	29	0.10
gender	22	0.04	gender	28	0.06	device name	29	0.10
birth	16	0.03	device ID	26	0.06	audio	27	0.09
device ID	7	0.01	birth	19	0.04	device ID	20	0.07
browsing history	7	0.01	user ID	14	0.03	user ID	7	0.02
user ID	4	0.00	Android ID	8	0.01	search history	3	0.01
search history	2	0.00	IMEI number	6	0.01	browsing history	3	0.01
IMEI number	1	0.00	frequency of use	6	0.01	IMEI number	1	0.00
Android ID	1	0.00	IMEI number	4	0.00	frequency of use	1	0.00
frequency of use	0	0.00	search history	2	0.00	Android ID	0	0.00

most commonly leaked to app providers. Other frequently accessed private attributes include address, email, and location. Across all six AI app categories, these private attributes are ranked in the top five in terms of the number and ratio of AI apps that accessed them. Moreover, we found that, for different categories of AI apps, the private attributes that are most easily leaked are similar (e.g., name, address, and email).

**Finding 19:** The most frequently accessed private attributes by the collected AI apps are name, address, email, and location.

Table 7 presents the privacy issues highlighted in user comments. The table categorizes these privacy issues into five main categories, providing detailed descriptions along with actual user comments as examples. Firstly, we see that privacy invasion and data misuse are among the top concerns. Users are worried about certain applications requesting excessive permissions that are unrelated to their core functions or even accessing device resources without authorization. This behavior makes users feel that their privacy is being violated, particularly when sensitive permissions such as camera and microphone access are accessed.

Secondly, users are worried about the lack of transparency in privacy policies. Many users find that privacy policies are difficult to access or understand, and they are

concerned that these policies may be changed without notification. Another major concern for users is the issue of third-party data sharing. Users complain that some applications share their data with third parties without their explicit consent. Lastly, the trade-off between privacy and user experience is another significant issue. Users report that some applications. For example, users may be forced to accept privacy terms in order to continue using the app, and this compulsory approach has resulted in user dissatisfaction.

**Finding 20:** The main privacy-related concerns raised by users include: privacy invasion and data misuse, lack of transparency in privacy policies, third-party data sharing, privacy protection features, and the balance between privacy and user experience.

## 6.2. RQ6: Analysis of user reviews related to AI technology

### 6.2.1. Objectives

User reviews are crucial for improving the AI techniques used in AI apps and enhancing user satisfaction. They can help identify potential issues with current AI techniques utilized in AI apps, contributing to the overall quality and reliability of the apps. In this research question, we investigate users' attitudes toward AI techniques in AI apps. Understanding users' attitudes can help developers identify current issues that users perceive in AI apps and create AI apps that better meet user expectations.

### 6.2.2. Experimental methodology

In the first step, we conducted web crawling from the Google Play application market to retrieve available reviews associated with our collected AI apps. Since in our collected dataset of AI apps, 88.7% of the AI apps belong to Google Play, we chose Google Play as the primary source for collecting reviews. In the second step, we constructed an AI-relevant technical keyword dictionary to filter out reviews that specifically focused on AI techniques. Based on the filtering, we obtained a collection of technical reviews (i.e., reviews regarding AI techniques).

For the collected review data, we performed three types of analyses: **1) Overall sentiment analysis**, where we employed a Transformers-based model (Wolf, Debut, Sanh, Chaumond, Delangue, Moi, Cistac, Rault, Louf, Funtowicz, Davison, Shleifer, von Platen, Ma, Jernite, Plu, Xu, Scao, Gugger, Drame, Lhoest and Rush (2020)) renowned for its effectiveness in sentiment classification tasks, achieving an impressive accuracy rate of 91.3%. Each technical review was classified as either positive or negative by the Transformers-based model. Moreover, we conducted a deeper analysis of the reasons for the negative reviews. We manually checked all the negative reviews and categorized the reasons for the negativity. **2) Strengths and weaknesses analysis**, involving manual examination and summarization of the identified technical reviews' notable attributes and limitations; **3) Categorized sentiment analysis**, wherein we



**Table 7**  
Privacy concerns highlighted in user comments

Main Privacy Concerns	Details	User Comment Examples
Privacy Invasion & Data Misuse	Users are concerned about apps requesting excessive permissions unrelated to their core functions, and unauthorized access to device resources.	"To use video chat with a doctor, I had to agree to invasive permissions like accessing my camera, microphone, calendar, and more." "This app accessed my microphone in the background even after I revoked permission. Who knows how long it's been recording me? Major invasion of privacy!"
Lack of Transparency in Privacy Policies	Privacy policies are hard to access or understand, with changes made without user notification.	"If I can't read their privacy agreements, I can't use the app." "The privacy policy is not acceptable. Because it might be changed without informing the users, as it mentions in the roles of the application!"
Third-Party Data Sharing	Apps shares data with third parties, especially without explicit consent.	"App worked well. Until today when I was asked if would agree to a third party privacy advertising policy. I chose option 'No thank you' but instead of being allowed to proceed, it merely removed the option I had chose and would allow me to continue in app until I chose to agree." "This app collects WAY too much data than should be needed to use the service, this app also collects it to use toward advertising."
Privacy Protection Features	Some users appreciate apps that offer privacy protection, like encrypted communication.	"This app actually respects your privacy." "Connect to people without disclosing privacy."
Privacy vs. User Experience	Users report a trade-off between privacy protection and user experience, with some apps limiting functionality unless privacy terms are accepted.	"Now I've disabled the annoying adds I now have to agree to a privacy policy every time I open the app??? Why can't it remember I have clicked accept?" "Trying to sign up, but can't get past ticking the terms and conditions and privacy policy box. No option to complete the action after ticking box."

computed the ratio of positive and negative reviews for each app category, facilitating further exploration.

### 6.2.3. Results

The experimental results for **RQ6** are presented in Figure 13, Figure 14, Table 8, Table 9, and Figure 15. Figure 13 illustrates the proportion of positive and negative reviews among all the collected technical reviews using a pie chart. We can see that positive reviews account for 68.4% of the total, while negative reviews constitute 31.6%. Based on the experimental results, we find that for the techniques used in AI apps, the proportion of positive reviews is higher compared to negative reviews, indicating that users' attitudes towards the techniques in AI apps are relatively positive.

Figure 14 shows the main reasons causing negative reviews (as mentioned above, 31.6% of reviews are negative). Below, we explain each of these main reasons.

- **Accuracy Issues** refer to the AI applications making inaccurate classifications, such as mistakenly identifying a hamburger as a hot dog.
- **Incomplete Functionality** refers to the AI applications lacking some expected functionality. For instance, in the "Photography" category, a user mentioned that the face grouping feature did not work properly.
- **Crashes and Errors** refer to instances where AI applications crash or generate error messages. For example, in the "Libraries & Demo" category, a user mentioned, "When I click the Recognition Test Button, the app stops working."
- **Dependency on External Libraries** refers to the application needing certain external libraries to function properly. For instance, in the "Video Players & Editors" category, a

user commented that the app required OpenCV Manager to run, a library that was not available in the app store.

- **Lack of Personality** refers to the app's lack of personalized features, failing to meet users' needs for a personalized experience. For instance, in the "Education" category, a user commented that while the app was powerful, it lacked personalized elements during use.
- **Key Features Missing** refers to the AI application lacking critical features compared to other similar apps. For example, in the "Finance" category, a user mentioned that the app lacked fingerprint authorization and NFC payment, which led them to consider other apps.
- **Slow Performance** refers to the AI application running slowly, affecting the user experience. For example, in the "Libraries & Demo" category, a user mentioned that the object detection speed of the app was not fast enough to meet real-time requirements.
- **Complex/Unfriendly Interface Issues** refer to instances where the AI app has a complicated or user-unfriendly interface, making it difficult for users to operate. For example, in the "Productivity" category, users reported that the app's interface did not refresh after deleting a file, which made it challenging to use.

From Figure 14, we see that Accuracy Issues are the leading cause of negative user reviews, accounting for 22.6% of all negative feedback. Following this, Incomplete Functionality accounts for 17.7%, which is the second most significant cause. Crashes and Errors are the third major problem, comprising 16.1% of the negative reviews. From

the above analysis, we conclude that accuracy issues, incomplete functionality, and crashes and errors are the three main sources of negative reviews.

**Finding 21:** AI app users' attitudes towards the techniques used in AI apps are relatively positive. Accuracy issues, incomplete functionality, and crashes and errors are the three main sources of negative reviews.

Table 9 presents summaries for some example technical reviews across different AI app categories. We can see that, 73.7% of reviews are completely positive, without pointing out any shortcomings. The remaining reviews are partly negative, highlighting advantages but pointing out issues. For example, for Travel & Local-oriented AI apps, users emphasize the limitations in translating US English to Hebrew. Among the negative reviews, 75% focus on accuracy issues (e.g., the prediction accuracy is insufficient), including in the Casual, Dating, Entertainment, and Photography categories of apps.

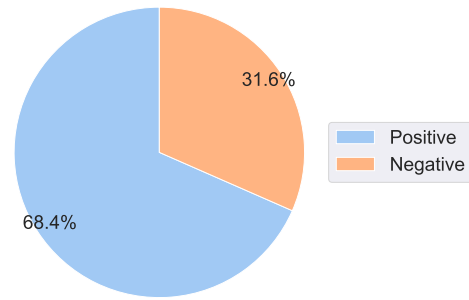
**Finding 22:** Among the negative reviews of AI technology in AI apps, the most frequently reported issue by users is accuracy issues (e.g., the prediction accuracy is insufficient).

Figure 15 depicts the proportion of positive and negative technical reviews across different categories. Notably, in more than half of the categories, the proportion of positive exceeds that of negative reviews. For instance, in the categories of Maps & Navigation, Parenting, Education, and Social, the ratio of positive reviews surpasses 90%. However, certain categories exhibit a higher proportion of negative reviews, such as Book & Reference, Entertainment, and Casual.

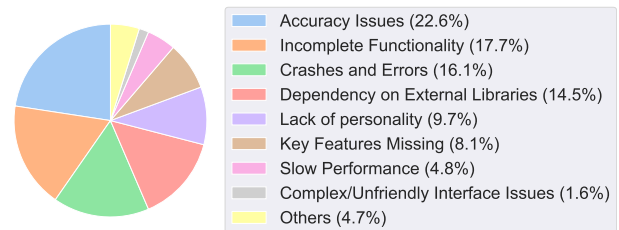
Table 8 presents the causes of negative reviews across categories with higher negative sentiments. We see that, in the **Entertainment** category, users' negative reviews mainly focus on recognition accuracy issues, such as errors in object recognition. For example, identifying a door as a refrigerator or a desk as a microwave. In the **Casual** category, users' negative reviews mainly focus on inaccuracies in image classification, such as identifying a hamburger as a hot-dog. Additionally, users suggested some functionality issues, such as adding confidence measures for algorithms. In the **Health & Fitness** category, users were concerned about technical details, particularly regarding the machine learning modules used.

The negative reviews of the **Dating** category were mainly caused by algorithm accuracy and the lack of personalization. Negative feedback in the **Lifestyle** category focused on the dependency on external components, such as the requirement to install OpenCV Manager. In the **Travel & Local** category, users mainly reported issues with voice recognition and translation, especially between American English and Hebrew. Lastly, negative reviews for **Books & Reference** apps primarily focused on technical issues, such as crashes or failures to start the app.

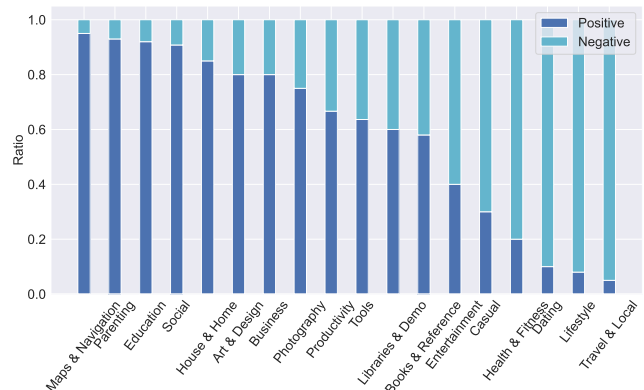
**Finding 23:** In more than half of the AI app categories, the proportion of positive exceeds that of negative reviews.



**Figure 13:** Sentiment analysis of technical reviews for AI apps: Proportion of positive and negative reviews



**Figure 14:** Analysis of causes for negative reviews of AI applications



**Figure 15:** Sentiment analysis results on AI technical reviews across different categories

## 7. Discussion

### 7.1. Challenges and Opportunities

In this section, we discuss the challenges and opportunities of deploying AI technologies to mobile applications.

#### 7.1.1. Challenges

In the following, we elaborate on the challenges of deploying AI technologies to mobile applications from two perspectives: on-device deployment and on-cloud deployment.

**Table 8**  
Causes of negative reviews across categories with higher negative sentiments

Category	Issues	Detailed Description	Examples from Reviews
Entertainment	Accuracy issues	Users mentioned poor object detection and recognition accuracy, such as recognizing a door as a refrigerator or a desk as a microwave, which obviously affects user experience.	"Recognizing a door as a refrigerator, a desk as a microwave"
Casual	Misclassification, Functionality issues	Users mentioned inaccurate image classification, such as misclassifying a hamburger as a hotdog. Users suggested adding confidence measures for algorithms and using front-facing cameras, which would improve the app's practicality.	"Misclassifying a hamburger as a hotdog, lack of front-facing camera support"
Health & Fitness	Technical details	Users inquired about the specific machine learning modules used but did not receive satisfactory answers, indicating a lack of technical transparency.	"Questions about TensorFlow usage not satisfactorily answered"
Dating	Algorithm accuracy, Lack of personalization	Users doubted the accuracy of facial recognition and suggested adding user verification for algorithm accuracy. Users expected more personalized and interactive features.	"Facial recognition accuracy doubted, lack of user verification"
Lifestyle	Dependency on external components	Users mentioned the need to install OpenCV Manager, which could be difficult to find or install in some cases, leading to the app not functioning properly	"Difficulty finding or installing OpenCV Manager"
Travel & Local	Voice recognition and translation issues	Users reported that the app's voice model had difficulty with American English and produced inaccurate translations, especially from English to Hebrew. They noted that other similar voice assistants (like Google and Alexa) did not have these issues.	"Poor recognition of American English, inaccurate Hebrew translations"
Books & Reference	Technical issues	Some users reported crashes or failure to start, such as inability to download necessary opencv packages.	"Inability to download opencv packages"

**Table 9**  
Summaries of technically-relevant reviews of AI apps

Category	AI Review summary
Art & Design	Good application to create paintings from photos and powerful application created using machine learning.
Books & Reference	Very good scikit-learn documentation.
Business	The scanning is good.
Casual	Useful but not always accurate.
Dating	It would be better for machine learning if the users could verify its accuracy.
Education	It covers all machine learning techniques and helps us understand the basics of machine learning, which is excellent.
Entertainment	It is an interesting app to learn about machine learning, but the accuracy is not good.
Health & Fitness	Which machine learning Module have you used for your predictive analysis.
House & Home	It contains object detection and is very excellent.
Libraries & Demo	Great app, very helpful for exploring deep learning.
Lifestyle	Wants to install OpenCV Manager.
Maps & Navigation	Use AI and deep learning to detect farm plots. A very useful and light app.
Parenting	Use machine learning to detect explicit images on Reddit or Google.
Photography	Object detection is not very accurate, but it was fun.
Productivity	Very good text recognition app.
Social	It can be used in various advanced scenes and balance modes, such as using voice to invoke the camera.
Tools	Text Recognition is an exciting and fantastic application with outstanding new features.
Travel & Local	Spanish translation is perfect but abysmal results for translating US English to Hebrew.

One of the primary challenges in on-device AI deployment is the limitation of computational resources. Mobile devices have limited computational power, battery life, and storage capacity. Running AI models on such constrained devices can lead to performance issues, such as slow processing times, thus impacting the user experience. Another crucial challenge lies in model protection. As the models are deployed locally, the risk of model theft increases, which can result in intellectual property loss. In RQ4 of our empirical

study (cf. Section 5.2), we show that several embedded models in AI apps are not encrypted, indicating the risk of model theft.

One of the primary challenges in on-cloud AI deployment is data privacy and security. AI systems typically require access to large amounts of user data, which can include sensitive information. Since AI models are deployed in the cloud, user data is stored on remote servers managed by third-party providers, leading to potential privacy and security issues. Another critical challenge lies in network dependency. Cloud-based processing requires a stable and high-speed internet connection. Poor connectivity can lead to a degraded user experience.

Some common challenges for both on-device and on-cloud deployment include model accuracy issues. In RQ6 of our empirical study, we find that 80% of negative reviews highlight model accuracy issues, considering that the predictions of the models are not accurate enough. Another critical challenge is user privacy concerns. Our experimental results in RQ5 demonstrate that most AI apps can access several crucial attributes of sensitive user data (e.g., name, address, email, images, etc.), which increases the risk of user-sensitive information leaks.

### 7.1.2. Opportunities

Deploying AI technologies in mobile applications offers numerous opportunities. First, AI techniques enhance the user experience through personalization. For instance, AI-driven personal assistants can learn from individual interactions to better predict user needs and offer more relevant assistance. Moreover, in some crucial fields like health monitoring and financial services, AI technologies can enable mobile applications to process and interpret complex data

in real time. For example, mobile health apps can utilize AI technologies to provide real-time analysis of health data, preventing users from potential health issues before they become critical.

## 7.2. Future Directions

Though AI techniques have demonstrated significant potential in enhancing mobile applications, several challenges remain due to the unique constraints and requirements of mobile environments. In this section, we suggest five future directions for AI in mobile applications.

- **Exploring LLM-Driven AI Applications** With the growing capabilities of large language models (LLMs), there is a significant opportunity to integrate LLM models into mobile applications to enhance user interaction and provide more advanced features. By integrating LLMs, mobile apps can offer more personalized experiences, enabling sophisticated features like real-time language translation and intelligent content generation. Exploring LLM-driven AI applications is a valuable future direction.
- **User Privacy Protection** As mobile applications increasingly rely on AI to process personal data, ensuring privacy and security becomes crucial (Zhang, Patras and Haddadi (2019)). This raises future research regarding ensuring data privacy and security without compromising the performance and usability of AI models.
- **AI Model Security** AI models are vulnerable to adversarial attacks, where small, deliberate perturbations to the input data can result in incorrect outputs. This vulnerability can severely affect the reliability of AI applications, particularly in critical fields like healthcare, finance, and autonomous driving. How to protect AI models from such attacks could be a future research direction.
- **Optimization of Lightweight Models** AI on mobile platforms requires balancing performance with limited computational and battery resources. While techniques like model compression and pruning strive to achieve this, maintaining optimal performance with minimal reduction in accuracy remains a challenge. This raises the question of how to design AI models that deliver high accuracy while being computationally lightweight on mobile devices.
- **Real-time Processing** The demand for real-time AI processing on mobile devices is growing, particularly in applications like augmented reality, real-time translation, and autonomous navigation (Battineni, Chintalapudi, Ricci, Ruocco and Amenta (2024), Omar and Salih (2024)). Achieving real-time performance with limited hardware resources is challenging. How to enable real-time AI capabilities while maintaining high accuracy and low power consumption is a crucial question to address.

## 7.3. Recommendations for AI App Developers, Users and R&D

Based on the findings from the experimental results (cf. Sections 4.3 to Section 6.2), we discuss the concrete recommendations for AI application developers, AI users, and AI R&D.

For AI app developers: **1) Encrypt AI Models** Finding 18 reported that among all the collected AI models, only 520 models, accounting for just 0.25%, are considered to be encrypted. Therefore, we recommend developers prioritize encrypting models, especially those handling sensitive tasks, to protect against potential breaches. **2) Enhance User Experience with Accurate AI Models** Finding 21 reported the accuracy problem of AI models in AI applications. Specifically, accuracy issues are the most commonly reported problem in negative user reviews. We recommend that developers focus on improving the predictive accuracy of AI applications to enhance the user experience. **3) Optimize AI App Size and Performance** Finding 12 indicated that AI apps supported by deep learning tend to be larger in size. This insight suggests developers to consider focusing on optimizing app size to reduce download time and improve user experience.

For AI app users: **1) Provide Feedback to Improve AI Apps** User comments can provide valuable insights for improving AI apps. For example, Finding 21 mentions that accuracy issues, incomplete functionality, and crashes and errors are the three main sources of negative reviews, which can help developers identify the shortcomings of current AI apps and make targeted improvements. Therefore, we recommend that users provide more suggestions to help developers better improve AI apps. **2) Manage Privacy Settings Proactively** Finding 19 indicates that AI applications frequently access sensitive user data, such as names, addresses, emails, and locations. Therefore, we recommend that users regularly review and manage the privacy settings of AI applications. This can effectively reduce the risk of potential data breaches and protect users' privacy.

For AI R&D: **1) Analyze High-Scoring AI Apps** According to Finding 7, AI applications in the Racing Games and Comics categories have the highest average scores. We recommend analyzing the factors contributing to the high ratings of these applications and applying these successful strategies to the development of AI applications in other categories to enhance overall app quality and user ratings. **2) Continued Focus on User Feedback** Findings 20, 21, and 22 highlight user feedback on current AI apps. We recommend that AI R&D teams continue to focus on user feedback, particularly on negative feedback, and use it as a crucial reference for optimizing applications. This approach will further enhance user experience and satisfaction.

## 7.4. Threats to Validity

*Threats to External Validity.* The external threat of the study lies in the model protection measurement method we applied in RQ4. We adopted the standard entropy test approach to determine whether a model is encrypted, which

assesses the encryption status of an AI model based on the model's entropy value. If the value exceeds the threshold, the model is considered encrypted. However, the accuracy of the method is not guaranteed. To mitigate this threat, we adopt the threshold value of 7.99 since it has been validated by the work of Sun *et al.* Moreover, we validate the accuracy of the approach standard entropy test by manually checking. Specifically, we randomly selected 50 AI application models considered encrypted by the standard entropy test and manually verified whether they were actually protected through two steps. First, we tried to open them using the Netron viewer, but none of the models could be opened. Subsequently, we attempted to use the official API of these AI models to load them, but we also found that none of the models could be opened.

*Threats to Internal Validity.* A major threat to internal validity arises from the manual collection of keywords for identifying AI applications. We collected a set of AI-related keywords to distinguish AI apps from the mobile apps in AndroZoo. To mitigate this threat, we invested sufficient time in gathering the keywords by consulting extensive literature and related GitHub materials. Moreover, as some simple AI terms (e.g., LSTM, CNN) can also appear in non-AI words/phrases, we excluded them from the keyword dictionary to improve the accuracy of AI app identification. Additionally, recognizing that some AI apps do not rely on AI frameworks and thus may not be detected using framework keywords, we supplemented the keyword dictionary with package names of ML/DL algorithms.

## 8. Related Work

### 8.1. Mobile Deep Learning

Deploying deep learning (DL) techniques to mobile devices has shown remarkable benefits, including quick response time, network independence, and enhanced privacy protection, thus attracting much attention in recent studies (Cheng, Wang, Zhou and Zhang (2017); Huang and Chen (2022); He, Lin, Liu, Wang, Li and Han (2018); Xu *et al.* (2019); Zhang, Zhou, Lin and Sun (2018); Howard, Zhu, Chen, Kalenichenko, Wang, Weyand, Andreetto and Adam (2017)). Cheng *et al.* (2017) conducted a comprehensive review of state-of-the-art techniques for compressing DNN models, including parameter pruning and quantization, low-rank factorization, compact convolutional filters, and knowledge distillation. He *et al.* (2018) proposed an effective model compression tool, AutoML, which utilized reinforcement learning to sample the design space and improve model compression quality. Xu *et al.* (2019) conducted the first large-scale study to explore the development progress of on-device deep learning and contributed valuable new findings. For example, early adopters of deep mobile learning are the top applications where embedded deep learning technology plays an important role. Zhang *et al.* (2018) introduced a computation-efficient CNN architecture, ShuffleNet, especially for mobile devices with minimal computing power. Howard *et al.* (2017) demonstrated MobileNets, a new model

architecture for mobile and embedded vision applications that achieved significant performance compared to other popular models on ImageNet classification.

### 8.2. ML/DL as cloud services

Unlike deploying ML/DL models directly to mobile devices, traditional computing paradigms prefer an online mode, where models are deployed on cloud platforms to perform training and inference. Under this mechanism, the mobile device sends data to the remote end and receives the prediction results. MLaaS (Machine Learning as a Service) (Ribeiro, Grolinger and Capretz (2015)) is a prevalent cloud service that offers a suite of pre-built machine learning tools and capabilities, allowing users to perform data analysis and prediction without needing to deeply understand the principles of ML algorithms. Yao, Xiao, Wang, Viswanath, Zheng and Zhao (2017) reviewed the effectiveness of MLaaS systems ranging from fully automated, turnkey systems to fully customizable systems, observing that user control can affect ML task performance. Shokri, Stronati, Song and Shmatikov (2017) empirically evaluated classification models trained by commercial MLaaS providers (e.g., Google and Amazon) from the perspective of model security, designing the first inference attack against ML models provided by Google Prediction API and Amazon. Tramèr, Zhang, Juels, Reiter and Ristenpart (2016) investigated the vulnerability of machine learning models offered by MLaaS providers to model extraction attacks. They demonstrated that adversaries with black-box access to these models can replicate them by making numerous queries to the prediction APIs, even when confidence scores are omitted. They underscore the insufficiency of removing confidence values as a protective measure and call for more robust security strategies for protecting these models.

## 9. Conclusion

In this paper, we conducted the most extensive empirical study on AI-driven applications, focusing on on-device ML apps, on-device DL apps, and AI service-supported apps. By analyzing 56,682 real-world AI applications identified from a pool of 7,259,232 mobile apps in the AndroZoo repository, we provide several key insights into the landscape of AI in mobile applications across three main perspectives: application analysis, framework and model analysis, and user analysis. For example, from the **application analysis** perspective, we find that incorporating AI technology into applications has become a growing trend since 2018, with the Finance and Business categories releasing the highest number of AI apps. From the **framework and model analysis** perspective, we find that AI apps supported by on-device DL techniques accounted for the highest proportion, and TFLite is the most prevalent framework among single-framework AI apps. From the **user analysis** perspective, we find that the most frequently accessed private attributes by the collected AI apps are name, address, email, and location, and users generally have a positive attitude towards AI in apps, with accuracy issues being the most reported problem

in negative reviews. Our detailed analysis offers insights into the prevalence, update practices, framework usage, and model protection in AI apps, guiding future AI app development and maintenance strategies. Moreover, by examining user privacy and attitudes, we highlight the importance of privacy protection in AI app development and offer insights into how users perceive current AI technologies utilized in AI apps. We provide a large-scale AI app dataset for further research, offering a valuable resource for the academic and developer communities.

## Acknowledgements

This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 949014).

## 10. Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## 11. Data Availability

We make our data and scripts publicly available at <https://zenodo.org/records/12205325>.

## References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., 2016. {TensorFlow}: a system for {Large-Scale} machine learning, in: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283.
- AI, A., 2023a. Alexa ai URL: <https://developer.amazon.com/en-US/alexa/>.
- AI, A., 2023b. Amazon ai URL: <https://aws.amazon.com/ai/>.
- AI, A., 2023c. Azure ai URL: <https://azure.microsoft.com/>.
- AI, G., 2023d. Google ai URL: <https://ai.google/>.
- Ali, A.H., 2023. Green ai for sustainability: leveraging machine learning to drive a circular economy. *Babylonian Journal of Artificial Intelligence* 2023, 15–16.
- Allix, K., Bissyandé, T.F., Klein, J., Le Traon, Y., 2016. Androzoo: Collecting millions of android apps for the research community, in: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), IEEE. pp. 468–471.
- Amos, B., Ludwiczuk, B., Satyanarayanan, M., et al., 2016. Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science* 6, 20.
- APKTool, 2023. Apktool URL: <https://github.com/iBotPeaches/Apktool/>.
- Battineni, G., Chintalapudi, N., Ricci, G., Ruocco, C., Amenta, F., 2024. Exploring the integration of artificial intelligence (ai) and augmented reality (ar) in maritime medicine. *Artificial Intelligence Review* 57, 100.
- Bilyk, Z.I., Shapovalov, Y.B., Shapovalov, V.B., Megalinska, A.P., Zhadan, S.O., Andruszkiewicz, F., Dołhańczuk-Śródka, A., Antonenko, P.D., 2020. Comparing google lens recognition accuracy with other plant recognition apps, in: Proceedings of the Symposium on Advances in Educational Technology, AET.
- Caffe, 2023. Caffe URL: <https://github.com/BVLC/caffe/>.
- Caffe2, 2023. Caffe2 URL: <https://github.com/facebookarchive/caffe2/>.
- Chainer, 2023. Chainer URL: <https://github.com/chainer/chainer/>.
- Chen, Z., Yao, H., Lou, Y., Cao, Y., Liu, Y., Wang, H., Liu, X., 2021. An empirical study on deployment faults of deep learning based mobile applications, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE. pp. 674–685.
- Cheng, Y., Wang, D., Zhou, P., Zhang, T., 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.
- CNNdroid, 2023. Cnndroid URL: <https://github.com/ENCP/CNNdroid/>.
- CNTK, 2023. Cntk URL: <https://github.com/microsoft/CNTK/>.
- Dang, X., Li, Y., Ma, W., Guo, Y., Hu, Q., Papadakis, M., Cordy, M., Traon, Y.L., 2024a. Towards exploring the limitations of test selection techniques on graph neural networks: An empirical study. *Empirical Software Engineering* 29, 112.
- Dang, X., Li, Y., Papadakis, M., Klein, J., Bissyandé, T.F., Le Traon, Y., 2023. Graphprior: mutation-based test input prioritization for graph neural networks. *ACM Transactions on Software Engineering and Methodology* 33, 1–40.
- Dang, X., Li, Y., Papadakis, M., Klein, J., Bissyandé, T.F., Le Traon, Y., 2024b. Test input prioritization for machine learning classifiers. *IEEE Transactions on Software Engineering*.
- Datumbbox, 2023. Datumbbox URL: <https://github.com/datumbbox/datumbbox-framework/>.
- David, R., Duke, J., Jain, A., Janapa Reddi, V., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Wang, T., et al., 2021. Tensorflow lite micro: Embedded machine learning for tinyml systems. *Proceedings of Machine Learning and Systems* 3, 800–811.
- DeepLearning4J, 2023. Deeplearning4j URL: <https://github.com/eclipse/deeplearning4j/>.
- Deng, Z., Chen, K., Meng, G., Zhang, X., Xu, K., Cheng, Y., 2022. Understanding real-world threats to deep learning models in android apps, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 785–799.
- Dospinescu, O., Popa, I., 2016. Face detection and face recognition in android mobile applications. *Informatica Economica* 20, 20.
- FeatherCNN, 2023. Feathercnn URL: <https://github.com/Tencent/FeatherCNN/>.
- Framework, N., 2023. Neuroph framework URL: <https://github.com/neuroph/NeurophFramework/>.
- Gamble, A., 2020. Artificial intelligence and mobile apps for mental health-care: a social informatics perspective. *Aslib Journal of Information Management* 72, 509–523.
- He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S., 2018. Amc: Automl for model compression and acceleration on mobile devices, in: Proceedings of the European conference on computer vision (ECCV), pp. 784–800.
- Hjelmås, E., Low, B.K., 2001. Face detection: A survey. *Computer vision and image understanding* 83, 236–274.
- Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- Huang, Y., Chen, C., 2022. Smart app attack: hacking deep learning models in android apps. *IEEE Transactions on Information Forensics and Security* 17, 1827–1840.
- Huang, Y., Hu, H., Chen, C., 2021. Robustness of on-device models: Adversarial attack to deep learning models on android apps, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), IEEE. pp. 101–110.
- Hub, T.L., 2023. Tensorflow lite hub URL: <https://tfhub.dev/s?deployment-format=lite&subtype=module,placeholder/>.
- Jones, H.G., 2020. What plant is that? tests of automated image recognition apps for plant identification on plants from the british flora. *AoB Plants* 12, plaa052.

- Joshi, R.D., Dhakal, C.K., 2021. Predicting type 2 diabetes using logistic regression and machine learning approaches. *International journal of environmental research and public health* 18, 7346.
- Keras, 2023. Keras URL: <https://github.com/keras-team/keras/>.
- Kindylidi, I., Cabral, T.S., 2021. Sustainability of ai: The case of provision of information to consumers. *Sustainability* 13, 12064.
- LaValley, M.P., 2008. Logistic regression. *Circulation* 117, 2395–2399.
- Li, Y., Dang, X., Ma, L., Klein, J., Traon, Y.L., Bissyandé, T.F., 2023. Test input prioritization for 3d point clouds. *ACM Transactions on Software Engineering and Methodology*.
- Li, Y., Dang, X., Pian, W., Habib, A., Klein, J., Bissyandé, T., 2024. Test input prioritization for graph neural networks. *IEEE Transactions on Software Engineering*.
- Li, Y., Hua, J., Wang, H., Chen, C., Liu, Y., 2021. Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE. pp. 263–274.
- Lite, P., 2023. Paddle lite URL: <https://github.com/PaddlePaddle/Paddle-Lite/>.
- Locke, S., Bashall, A., Al-Adely, S., Moore, J., Wilson, A., Kitchen, G.B., 2021. Natural language processing in medicine: a review. *Trends in Anaesthesia and Critical Care* 38, 4–9.
- Lu, J., Wu, D., Mao, M., Wang, W., Zhang, G., 2015. Recommender system application developments: a survey. *Decision Support Systems* 74, 12–32.
- MACE, 2023. Mace URL: <https://github.com/XiaoMi/mace/>.
- MALLET, 2023. Mallet URL: <https://github.com/mimno/Mallet/>.
- Matarneh, R., Maksymova, S., Lyashenko, V., Belova, N., 2017. Speech recognition systems: A comparative review.
- Miner, R., 2023. Rapid miner URL: <https://rapidminer.com/>.
- MLPACK, 2023. Mlpack URL: <https://github.com/mlpack/mlpack/>.
- Morawiec, D., 2021. sklearn-porter. URL: <https://github.com/nok/sklearn-porter>. transpile trained scikit-learn estimators to C, Java, JavaScript and others.
- NCNN, 2023. Ncnn URL: <https://github.com/Tencent/ncnn/>.
- NLP, B., 2023. Baidu nlp URL: <https://ai.baidu.com/ai-doc/NLP/>.
- OCR, B., 2023. Baidu ocr URL: <https://ai.baidu.com/ai-doc/OCR/>.
- Omar, L.I., Salih, A.A., 2024. Systematic review of english/arabic machine translation postediting: Implications for ai application in translation research and pedagogy, in: *Informatics*, MDPI. p. 23.
- OpenCV, 2023. Opencv URL: <https://github.com/opencv/opencv/>.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32, 8026–8037.
- Pham, V., Bluche, T., Kermorvant, C., Louradour, J., 2014. Dropout improves recurrent neural networks for handwriting recognition, in: 2014 14th international conference on frontiers in handwriting recognition, IEEE. pp. 285–290.
- Ribeiro, M., Grolinger, K., Capretz, M.A., 2015. Mlaas: Machine learning as a service, in: 2015 IEEE 14th international conference on machine learning and applications (ICMLA), IEEE. pp. 896–902.
- Rokach, L., Maimon, O., 2005. Decision trees. *Data mining and knowledge discovery handbook*, 165–192.
- Searcher, T.S., 2023. The silver searcher URL: [https://github.com/ggreer/the\\_silver\\_searcher/](https://github.com/ggreer/the_silver_searcher/).
- Shogun, 2023. Shogun URL: <https://github.com/shogun-toolbox/shogun/>.
- Shokri, R., Stronati, M., Song, C., Shmatikov, V., 2017. Membership inference attacks against machine learning models, in: 2017 IEEE symposium on security and privacy (SP), IEEE. pp. 3–18.
- SNPE, 2023. Snpe URL: <https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/>.
- Sun, Z., Sun, R., Lu, L., Mislove, A., 2021. Mind your weight (s): A large-scale study on insufficient machine learning model protection in mobile apps, in: 30th {USENIX} Security Symposium ({USENIX} Security 21).
- synthesizer, 2023. Synthesizer URL: <https://ai.baidu.com/tech/speech/>.
- Thakkar, M., Thakkar, M., 2019. Introduction to core ml framework. *Beginning Machine Learning in iOS: CoreML Framework*, 15–49.
- Tramèr, F., Zhang, F., Juels, A., Reiter, M.K., Ristenpart, T., 2016. Stealing machine learning models via prediction {APIs}, in: 25th USENIX security symposium (USENIX Security 16), pp. 601–618.
- TVM, 2023. Tvm URL: <https://github.com/apache/tvm/>.
- Wang, H., Wang, N., Yeung, D.Y., 2015. Collaborative deep learning for recommender systems, in: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1235–1244.
- WEKA, 2023. Weka URL: <https://github.com/ishaanjav/Weka-ML-Face-Recognition/>.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M., 2020. Transformers: State-of-the-art natural language processing, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, Online. pp. 38–45. URL: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.
- Xu, D., Tian, Y., 2015. A comprehensive survey of clustering algorithms. *Annals of data science* 2, 165–193.
- Xu, M., Liu, J., Liu, Y., Lin, F.X., Liu, Y., Liu, X., 2019. A first look at deep learning apps on smartphones, in: *The World Wide Web Conference*, pp. 2125–2136.
- Yao, Y., Xiao, Z., Wang, B., Viswanath, B., Zheng, H., Zhao, B.Y., 2017. Complexity vs. performance: empirical analysis of machine learning as a service, in: *Proceedings of the 2017 Internet Measurement Conference*, pp. 384–397.
- Zhang, C., Patras, P., Haddadi, H., 2019. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials* 21, 2224–2287.
- Zhang, X., Zhou, X., Lin, M., Sun, J., 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Zhao, N., Wu, M., Chen, J., 2017. Android-based mobile educational platform for speech signal processing. *International Journal of Electrical Engineering Education* 54, 3–16.