Automated Mobile Content Compliance Verification Using Multimodal Learning

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney

Dishanika Dewani Denipitiyage
Networked Systems and Security (NSS) Research Lab
2026

Declaration of Authorship

I, Dishanika Dewani Denipitiyage, declare that this thesis titled *'Automated mobile content compliance verification using multimodal learning'* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Abstract

In this digitalised world, inappropriate content is a major concern across various domains, including media, mobile apps, and computer games, mainly for children. As GDPR, COPPA, and APPS are enforced, each industry has started standardising the content available to end users. Content/Age rating in mobile apps describes the minimum maturity level of content in apps. Yet the content available in the mobile app ecosystem presents a significant security risk. While the mobile app privacy and security violations, malware/ spam/ counterfeit detection, have been extensively studied in research, content compliance detection remains in the rudimentary stage.

This thesis investigates four perspectives: 1) app metamorphosis, which measures the fundamental evolution of apps via app metadata rather than following a typical incremental updates. 2) cross-modal contrastive learning to identify content rating non-compliance across Google Play Store, 3) cross-model rationale generation, which targets interpretable explanations of content rating descriptor violations in a cross-platform setting and 4) a unified framework that detects out-of-distribution samples by analysing class-wise ranking violations in model outputs without auxiliary outlier data,

In the first part of this thesis, we investigate metamorphosis events in mobile applications by developing a multi-modal similarity search pipeline capable of identifying substantial shifts in mobile apps across five years. Our methodology combines multi-modal embeddings through a majority-voting algorithm to establish robust cross-dataset app correspondences. Further, we quantitatively characterise how successful an app is progressing from 2018 to 2023 with a novel score function. This research pioneered the work in identifying the prevalence of content rating non-compliance in Google Play Store.

The second part explores automated detection of content rating violations in Android apps by developing a vision–language representation generation model that jointly analyses app descriptions and visual creatives. The method emphasises the significance of coupling content and style features in visual data and combines with text features, then the textual and visual representations are aligned through a cross-attention module to reduce the semantic gap in context-sensitive mobile app data. Due to the inherent ordinal structure in content ratings, we employ ListMLE loss to predict ratings for each app as a downstream task. Evaluated on 10,000 Play Store games, including “Teacher Approved” apps, 34.5% of teacher-approved apps and 45.7%

of malpractices flagged by our method were later observed as discontinued.

Motivated by the strong performance of our ordinal classification, we extend the ListMLE loss to capture the inherent order in standard image classifiers such as ResNet18. We further demonstrate that violations of the learned rank structure provide an effective signal for out-of-distribution detection. Based on this insight, we introduce RankOOD, a unified framework that detects OOD samples by directly analysing class-wise ranking patterns in model outputs, without requiring auxiliary outlier data during training. Our method achieves SOTA performance, reducing FPR95 by 4.3% relative to the strongest baseline on TinyImageNet for the near-OOD setting.

In the final part of this thesis, we address the underdeclaration of content rating by the developers and the absence of a unified content rating framework in the mobile app ecosystem. We leverage the rigorously regulated Apple App Store as an external reference point. To enable cross-platform knowledge transfer, we introduce a content-descriptor-based data generation pipeline that transforms app creatives and descriptions into structured question–answer pairs designed to capture content descriptors. Rather than predicting age ratings directly, we fine-tune a vision language model to identify the presence or absence of specific content rating descriptors, thereby producing explainable outputs. The training procedure incorporates a two-step training strategy: supervised fine-tuning on 3,000 Apple apps to establish baseline semantic grounding, followed by mistakes-oriented DPO training to correct misclassifications. Compared to our method, Qwen3 baseline shows a reduction in positive recall of 52.8% for binary classification and 39% for impact-based multi-class classification.

Acknowledgements

I would like to extend my sincere gratitude to Dr. Suranga Seneviratne, my primary supervisor, for his consistent guidance and mentorship throughout this significant academic undertaking. Your guidance and direction shaped this body of work into something that I am proud to stand by. My gratitude also goes to the valuable efforts from Dr. Sanjay Chawla, Dr. Anirban Mahanti, and Professor Aruna Seneviratne in their co-supervision. Their constructive feedback greatly enhanced the quality of the work. I thank the Australian Research Council (ARC) Discovery Project (DP220102520) for the grant that provided the invaluable financial support. I would further acknowledge the support provided by the University of Sydney Tuition Fee Scholarship and Faculty of Engineering Research Stipend Scholarship during my candidature period.

This journey in my academic life would not have been possible if not for the nurturing support of my parents and my sisters. Thank you for always believing in me to fly high. I also thank my brothers-in-law for holding the fort at home, giving me the reassurance to work on my dream oceans away.

List of Abbreviations

CR	Content Rating
CRD	Content Rating Descriptor
ACB	Australian Classification Board
IARC	International Age Rating Coalition
AI	Artificial Intelligence
DNN	Deep Neural Network
SSL	Self Supervised Learning
CL	Contrastive Learning
VLM	Vision Language Model
MLLMs	Multimodal Large Language Model
MLP	Multi-Layer Perceptron
LLM	Large Language Models
ID	In-Distribution
OOD	Out-of-Distribution
MSP	Maximum Softmax Probability
RPM	Rank Probability Matrix

List of Publications

Journals

1. **D. Denipitiyage**, B. Silva, S. Seneviratne, A. Mahanti, and A. Seneviratne, "Detecting and Characterising Mobile App Metamorphosis in Google Play Store", *IEEE Transactions on Mobile Computing (TMC)* , Volume 24, 2024, pp. 7489 - 7504.

Conferences

2. **D. Denipitiyage**, B. Silva, S. Seneviratne, A. Seneviratne, and S. Chawla, "A Vision-Language Approach with Cross Attention for Detecting Content Rating Malpractices in Android Applications", *IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (Trustcom)* 2025.

3. **D. Denipitiyage**, N. Karunanayake, S. Seneviratne, and S. Chawla, "RankOOD - Class Ranking-based Out-of-Distribution Detection", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 2026.

4. **D. Denipitiyage**, A. Seneviratne, and S. Seneviratne, "QwenSafe: Multimodal Content Rating Description Identification via Preference-Aligned VLMs", *ACM ASIA Conference on Computer and Communications Security (ACM ASIACCS)* 2026, **(Under Review)**.

Publications during the candidature not relevant to this thesis

5. C. De Alvis, **D. Denipitiyage**, S. Seneviratne., "Long-Tail Learning with Rebalanced Contrastive Loss", *Elsevier NeuroComputing*, Volume 657, 2025.

6. B.Silva, **D. Denipitiyage**, S. Seneviratne, A. Mahanti, and A. Seneviratne. "Entailment-Driven Privacy Policy Classification with LLMs", *IEEE Building a Secure & Empowered Cyberspace (Build- Sec)*, 2024.

7. B. Silva, **D. Denipitiyage**, A. Mahanti, A. Seneviratne, and S. Seneviratne, PrivPRISM: "Automatically Detecting Discrepancies Between Google Play Data Safety Declarations and Developer Privacy Policies", *Privacy Enhancing Technologies Symposium*, 2026. **(Under review)**

Authorship Attribution Statement

We present the authorship attribution for the published works from the previous section (List of Publications) included in the thesis chapters.

- Chapter 3 of this thesis comprises content published as [1] in the List of Publications.

  In [1], I investigate metamorphosis within the Android app ecosystem by analysing large-scale evolutionary transformations of mobile apps in Google Play Store.

- Chapter 4 of this thesis comprises content published as [2] in the List of Publications.

  In [2], I developed a scalable framework for detecting content-rating violations in Android games, enabling the systematic identification of rating malpractices and developer disguises within the Google Play ecosystem.

- Chapter 5 of this thesis is published as [4] in the List of Publications.

  In [4], I developed the idea of QwenSafe, designed the fine-tuning framework for content rating descriptor identification, conducted all experiments, and drafted the manuscript.

- Chapter 6 of this thesis is published as [3] in the List of Publications.

  In [3], I developed the framework of RankOOD for out-of-distribution detection, conducted all experiments, and drafted the manuscript.

I certify that the authorship attribution statements provided above are correct, and I have obtained permission from the other authors to include the published materials. As the lead author, and by convention in my research field, I have made the primary contribution to the publications. Additionally, I am the corresponding author for the publications included in this thesis.

Dishanika D. Denipitiyage Signature: Date:

As the supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Dr. Suranga Seneviratne Signature: Date:

Use of Generative Artificial Intelligence

During the preparation of this work, GPT-based models and the Gemini-2.5-Flash model were used via API access as part of the research workflow, as explicitly described in the relevant methodology sections. In limited cases, generative AI was used for grammatical refinement of the text; all such content was carefully reviewed to identify and correct potential errors, inaccuracies, or bias. I take full responsibility for the content of this manuscript and confirm that the work is original and that generative AI tools were used in accordance with institutional guidelines and policies.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The global reliance on mobile technology has grown substantially over the last decade, transforming smartphones into indispensable tools for communication, entertainment, education, commerce, and social interaction. The global mobile app market is dominated by the Google Play Store [1] and the Apple App Store [2], which together host approximately four million applications [3, 4]. Mobile device adoption has expanded beyond adults to include adolescents and young children, who increasingly use smartphones for activities such as gaming, video streaming, and educational activities. A 2025 report indicates that more than 60.42% of the global population owns a smart device [5], with mobile penetration rates exceeding 85% in many developed regions [6, 7, 8]. More significantly, mobile device usage has exploded among younger populations, with research demonstrating that children are accessing smartphones at increasingly early ages. In 2024, the Pew Research Center reported that approximately 95% of teenagers aged 13-17 own or have access to a smartphone [9], while other studies indicate that children as young as eight years old are regular mobile device users [10]. This trend underscores the importance of ensuring that the digital environments accessed by young audiences are both safe and age-appropriate.

Mobile applications, commonly referred to as apps, are a central component of the modern digital ecosystem. They have significantly reshaped human behaviour and institutional operations, enabling personalised services and continuous connectivity. The banking sector, for instance, has witnessed a marked shift toward mobile-first strategies, with financial institutions reporting that mobile applications now account for the majority of customer transactions [11]. Similarly, educational institutions have increasingly adopted mobile applications to facilitate remote learning and student engagement [12], while healthcare providers utilise specialised applications for telemedicine, patient monitoring, and health management [13]. However, this widespread adoption has also introduced a range of challenges, including security threats, privacy concerns, and various forms of misuse and regulatory non-compliance within mobile app ecosystems.

Mobile malware: The relatively permissive nature of mobile application ecosystems has facilitated the rapid spread of malware. Historically, mobile malware predominantly targeted Android smartphones, owing to the platform's open-source architecture and the ease of distributing applications through third-party marketplaces [14].

Although security mechanisms such as Google Play Protect [15] have detected and blocked a large number of malicious applications before their public release, advanced techniques, including dynamic code loading, incremental malicious updates, and app repackaging, have enabled malware to bypass detection and propagate through official app stores [16]. On iOS, jailbreaking provides a mechanism to bypass Apple's built-in restrictions, allowing the installation of unauthorised applications and system tweaks. Further, sophisticated spyware such as Pegasus and Predator have demonstrated that even a tightly controlled platform such as iOS is not immune to advanced mobile malware threats [16].

Mobile greyware and malpractices: Beyond mobile malware, several works have studied various malpractices and dubious app behaviours in app markets, such as spamming [17, 18], cloning [19, 20], counterfeits [21, 22], and ranking frauds [23, 24]. These works involve deceptive tactics to artificially boost an app's visibility and popularity using fake reviews and inflated downloads, while some apps attempt to mimic the branding, functionality, or identity of popular apps in order to attract users, generate advertising revenue, or spread malware. Although existing studies [23, 25, 21, 17] have proposed techniques to identify these fraudulent behaviours, in a highly competitive and profit-driven ecosystem, fraudsters and threat actors can come up with new techniques to circumvent mitigations deployed by app markets.

Mobile app privacy violations: While it is known that app developers use tracking libraries in both free and paid apps [26, 27], numerous studies have examined privacy violations arising when an app's actual behaviour deviates from its declared privacy policy or breaches data protection laws (e.g., GDPR [28], CCPA [29], and COPPA [30]). User surveys, app developer communications, and static code analyses have highlighted widespread under-reporting of data practices (e.g. data sharing and data collection) following the introduction of the Data Safety section (**cf.** Figure 1.3) for Android apps in 2022. These studies also reveal significant inconsistencies between declared data safety disclosures and actual app behaviour [31]. Further, this research area extends to efforts to simplify privacy policies through summarisation [32, 33], support user decision-making [34], answer user queries [35], and to build user-interactive privacy tools [36] with the advancements in Natural Language Processing (NLP).

Such findings and associated research have significantly advanced mobile app end-user safety, particularly in areas such as data protection, permission misuse, and privacy policy review. Nonetheless, these efforts were overwhelmingly concentrated on security and privacy risks. In contrast, comparatively limited attention has been directed towards mobile app content

compliance. **Content or age or maturity rating (CR)** specifies the minimum required maturity level based on the content available in an application. Violations of CR, where the declared rating does not accurately reflect the app's actual content, can expose younger users to inappropriate material and undermine the very purpose of rating systems intended to guide parents and guardians.

Content rating authorities: Android follows region-specific content-rating schemes [37] in which games distributed in Australia adopt the Australian Classification Board (ACB) ratings [38], while general-purpose apps rely on the International Age Rating Coalition (IARC) system [39]. Other jurisdictions map to their respective authorities; i.e., Pan European Game Information (PEGI) [40] governs ratings across Europe and the Middle East, the Entertainment Software Rating Board (ESRB) [41] administers classifications in North and South America, the Unterhaltungssoftware Selbstkontrolle (USK) [42] oversees ratings in Germany, and the Game Rating and Administration Committee (GRAC) [43] regulates content ratings in South Korea. In contrast, Apple employs a predominantly centralised rating framework across all regions [44] with the exception of selected regions(cf. Figure 1.2). Apple employs a proprietary content rating system comprising four age tiers (4+, 9+, 13+, and 17+) with region-specific labels where required. In 2025, Apple expanded this framework into a more granular five-tier scheme by retiring the 12+ and 17+ categories and introducing 13+, 16+, and 18+ age ratings. These schemes evaluate the presence of content rating descriptors (CRDs) such as mature themes, sexuality or nudity, violence, gambling elements, drug references, and strong language [44, 37]. Across both platforms, the initial classification is derived from developer-completed questionnaires provided during the app-submission process, with each marketplace providing its own rating form and interpretation [45].

CR label assignment: Unlike security- or privacy-related disclosures, which are routinely validated through static and dynamic analysis and data-flow inspection, content ratings rely primarily on developer self-reporting and are not systematically audited at the time of initial app release. In practice, verification of content appropriateness typically occurs reactively, following user complaints or reports of inappropriate content, despite the fact that such verification is technically feasible. The content rating challenges outlined above stem primarily from infrastructural dependencies and the prioritisation of monetisation within app markets. First, content ratings are largely determined by developer self-reporting, with limited verification during the initial app review process. Second, Android's reliance on region-specific classification frameworks results in the same application receiving different age ratings across countries. Finally, Apple's use of a proprietary rating scheme introduces cross-platform inconsistencies.

Even though new generative AI models appear to provide a possible means to analyse visual and textual content at scale, their naive use lacks explicit grounding in the legal logics that underpin formal age classification systems. This limitation is further exacerbated by the

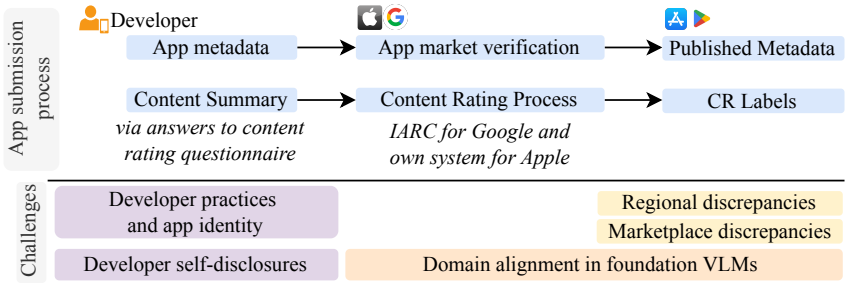


Figure 1.1: Overview of the mobile app submission process to app markets, highlighting key research gaps at each stage.

under-reporting of content ratings and descriptors by developers seeking to reach a broader audience, which restricts the availability of reliable, high-quality labeled data for model training. Moreover, the manual evaluation of individual apps required to construct larger datasets is both time-intensive and resource-intensive, further constraining the incorporation of recent vision language models (VLMs).

This thesis has identified the following research challenges that need to be addressed for a safer mobile app ecosystem. We summarise the identified research gaps in Figure 1.1, aligned with the three stages of the app submission process to the app market.

- **Understanding Developer Practices and Malpractices:** As discussed earlier, the mobile app ecosystem is highly dynamic, competitive, and profit-driven. Consequently, developers continuously seek ways to remain competitive, occasionally resorting to practices that violate app marketplace policies. To gain a comprehensive understanding of developer behaviour, it is therefore essential to study the long-term evolutionary patterns of mobile applications over extended periods. For example, Mafia City exhibits a transition in its content rating from PG to M, driven by the inclusion of violence and strong language. Such shifts raise significant safety concerns, as underage users may have already downloaded the app under PG rating. Similarly, shifts in hardware capabilities have prompted functional transformations within the Android ecosystem, with applications such as Noah Camera evolving into a photo editor app. Such longitudinal transformations are common within mobile app markets; however, whether and how these changes contribute to maintain user attraction, user safety and market success remains insufficiently understood.

- **Platform-Level Discrepancies and Use of Multiple Concurrent Rating Schemes:** Apple employs a proprietary classification scheme (4+, 9+, 13+, 17+) while also displaying geolocation-specific ratings for certain regions. In contrast, Android relies primarily on the IARC framework, which automatically assigns region-appropriate ratings based on user location. The use of these different rating infrastructures means that the same app can present multiple age ratings depending on platform and geography. The Canadian Centre for Child Protection illustrates this issue clearly: YouTube is labelled 17+ on Apple App Store, “Teen”

on Google Play (ESRB ratings), and 13+ in its own Terms of Service [46]. Figure 1.2 illustrates the distinct and overlapping content rating schemes used by the Android and iOS app marketplaces. For example, Apple defines three intermediate age categories (4+, 9+, and 12+) that span between [G, PG] and [M, MA15+] categories. Such disparities highlight the platform level discrepancies within mobile app ecosystems, which produce divergent interpretations of content suitability. These contradictions create a situation in which parents may encounter mutually inconsistent age thresholds for a single service, undermining the interpretability and reliability of platform-provided ratings.

- **Heavy Dependence on Questionnaire-Based Developer Self-Declarations:** Current app-rating pipelines rely predominantly on developer-submitted questionnaires that are translated into age or content rating labels [45, 37]. This reliance on self-declaration introduces an inherent vulnerability: developers may unintentionally misinterpret questions, underestimate the relevance of certain content, or strategically downplay sensitive features to broaden their potential user base. Since most marketplaces do not systematically verify the accuracy of these declarations through a manual review process or an automated content analysis, the risk of inaccurate ratings is amplified.

- **Regional Fragmentation and Non-Uniform Rating Standards:** Regional divergence in content-rating standards further complicates the reliability of mobile app classifications. Rating authorities across jurisdictions embed distinct cultural norms, regulatory expectations, and interpretive criteria, leading to heterogeneous assessments of the same application. This fragmentation is empirically evident: Sun et al. [47] report that 21.69% of the 9,453 apps they examined exhibited inconsistent content ratings across authorities. A representative example appears in the case of `com.my**in.app*`, a plant identification app with more than one million installs. While the IARC system labels it "PEGI 3" and "Rated for 3+", the rating changes to "USK 12+" and "ESRB Mature 17+" when the same app is viewed through the Play Store interface using `gl=de` and `gl=us` parameters. Such discrepancies demonstrate that the perceived suitability of an application depends not solely on its content but on the regional regulatory lens through which it is viewed. In the context of mobile ecosystems, where apps circulate globally and users routinely traverse markets, this variability introduces significant uncertainty for parents and policymakers attempting to understand content ratings.

- **Less Domain Alignment in Foundation VLMs to Assign Content Ratings in Mobile Apps:** Despite recent advances in multimodal generative AI models [48, 49] and vision–language models [50, 51], there remains a lack of reliable automated methods for assigning content ratings to mobile applications. Generic vision–language models, primarily trained on general-purpose web-scale image–text data, are not well aligned with the requirements of mobile

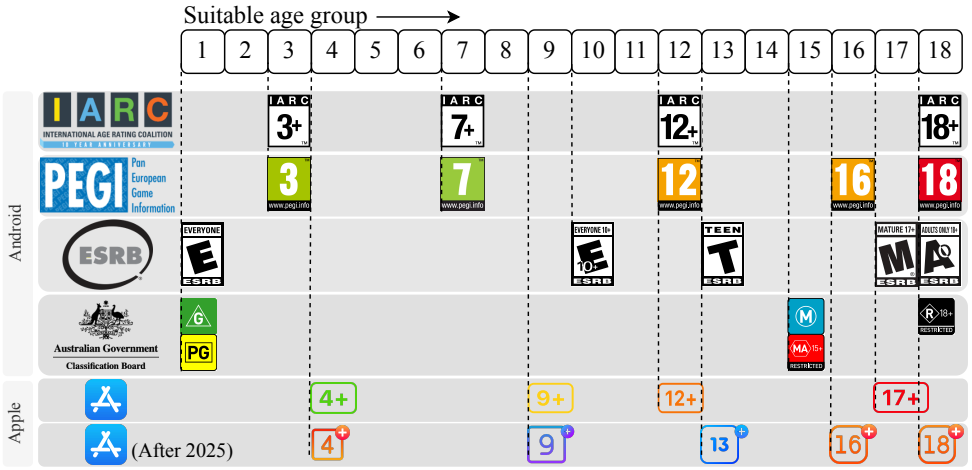


Figure 1.2: Comparison of age-based content rating categories across major rating authorities (IARC, PEGI, ESRB, ACB, and Apple App Store). The figure illustrates discrepancies in how apps are mapped to age groups across authorities, highlighting the lack of a unified standard for content ratings.

app content rating, which involves context sensitive semantics, regulatory criteria, and subtle safety cues. Effective adaptation of such models typically requires large-scale, domain-specific multimodal datasets [52], which are difficult to construct for mobile app markets due to the need for expert annotation and systematic identification of undeclared content. Consequently, existing approaches to mobile app content rating have predominantly relied on static and dynamic analysis techniques, including keyword-based methods [53, 54], feature-based models [55], and statistical approaches [47], rather than general-purpose VLMs.

These research challenges collectively undermine the reliability of age classifications and complicate efforts to ensure safe digital environments, particularly for younger audiences who rely on rating schemes as a primary layer of protection.

This motivates the central research question of this work: **How can we develop robust, scalable, and regulation-aware multimodal methods for accurate mobile app content rating under weak supervision, cross-platform discrepancies, and domain misalignment?** We focus on content-rating practices within the Australian context, where mobile applications are primarily governed by the Australian Classification Board (ACB) and regionally adapted interpretations of the IARC framework. To contextualise these challenges and establish the foundation for the analyses that follow, the next section introduces the key background concepts underpinning mobile app ecosystems and regulatory frameworks.

1.1 App Metadata and Content Rating Frameworks

Within mobile app ecosystems, regulatory and consumer-facing information is primarily communicated through "app metadata", which refers to the structured set of descriptive and declarative information that developers are required to submit during the app publication process and that platforms display to users, regulators, and other stakeholders. As shown in Figure 1.3, core metadata elements typically include the app name, app description, app genre, developer information, app screenshots, app icon, and age-related content rating information. Among these elements, content rating information occupies a distinctive role, as it serves simultaneously as a consumer guidance label and as a regulatory requirement.

Both Apple and the ACB used by Android employ hierarchical rating schemes, but their structuring and treatment of content differ substantially. As shown in Table 1.1, Apple organises its framework into seven high-level categories—In-App Controls, Capabilities, Mature Themes, Medical or Wellness, Sexuality or Nudity, Violence, and Chance-Based Activities—whereas ACB defines eight top-level domains, including Themes, Violence, Language, Drug Use, Nudity, Sex, Online Interactivity, and In-Game Purchases (**cf.** Appendix A.1). Although these first-layer categories appear broadly aligned, the second-layer descriptors, hereafter named as **Content Rating Descriptors (CRDs)**, diverge considerably and specify the nature of potentially sensitive or objectionable material present in an app. ACB uses fine-grained CRDs with impact levels spanning very mild, mild, moderate, strong, and high impact. Impact levels are modulated not only by frequency/ intensity, but also by contextual factors such as tone, purpose, realism, level of detail, interactivity, and whether content is depicted or merely implied. In contrast, Apple reduces intensity to a binary distinction: mild/infrequent versus intense/frequent, resulting in a coarser measurement scale. *These structural differences lead to significant inconsistencies in age-related decisions across the two systems*. Apple prohibits nearly all forms of sexual content below 13+, whereas ACB permits very mild sexual references, imagery, or non-sexual nudity within the G category. A similar pattern appears in violence: Apple disallows all violence in its 4+ tier, while ACB allows very mild cartoon, fantasy, and comedic violence in G. Apple also restricts horror/fear content to 9+, but ACB permits very mild horror or supernatural elements at G. Additional divergences include crude humour—prohibited in Apple's 4+ but allowed at very mild levels in ACB's G—and mature themes, which ACB permits at G but Apple reserves for 9+ (infrequent) or 13+ (frequent). Together, these discrepancies illustrate that *Apple's model is substantially more conservative at early age tiers, while ACB's multidimensional impact framework permits low-intensity material at younger classifications.*

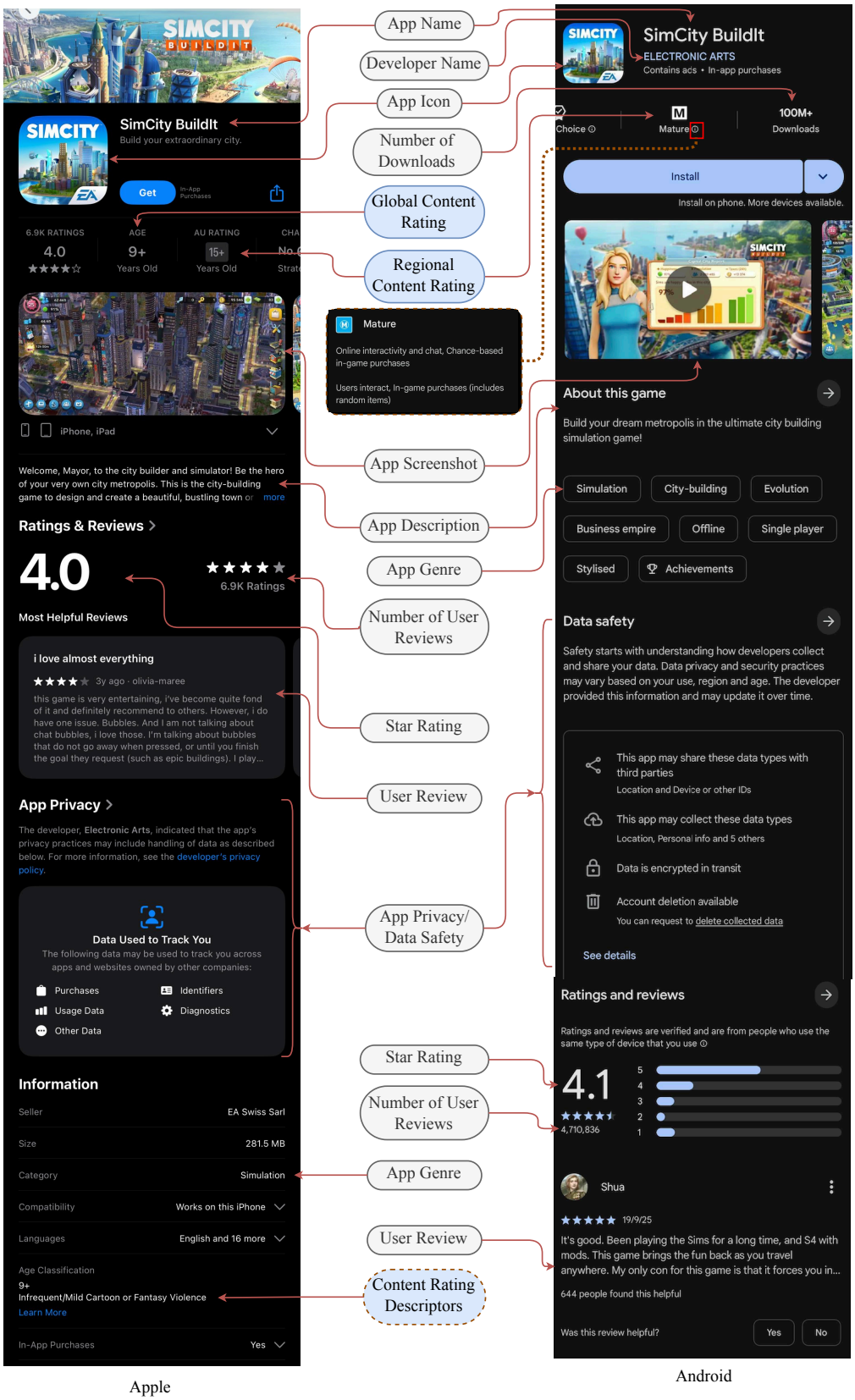


Figure 1.3: App metadata of Android and Apple apps

As outlined earlier, content ratings within mobile app metadata are generated through developer self-disclosure, which also varies by platform. The Google Play Store has adopted

the IARC framework as the foundation of its age rating system. Through direct integration of the IARC questionnaire into its developer dashboard, Google enables automated generation of region-specific age ratings that correspond to the requirements of participating authorities, including ESRB, PEGI, ACB and USK. To accommodate the scale of the mobile app ecosystem, IARC employs an automated questionnaire-driven pipeline in which developers self-report content descriptors and receive ratings instantaneously. The questionnaire encodes the distinct standards of each participating authority, using rule-based algorithms to generate region-appropriate ratings without the need for prior human review [39].

In contrast, the Apple App Store continues to rely on a proprietary content rating system that operates independently of IARC and national rating authorities. However, Apple also requires developers to self-report the presence and frequency of specific content descriptors through a platform-defined questionnaire, which then determines placement within Apple's internally defined age tiers [45]. The resulting classifications are not formally mapped onto established regional systems such as ESRB, PEGI, or ACB. As a result, *the same application may receive different content labels across the two dominant mobile platforms, even when distributed in the same jurisdiction.* Figure 1.3 illustrates a discrepancy in the content ratings assigned to SimCity BuildIt across platforms, with the app receiving a Mature (M, 15+) rating under the ACB authority on Android; however, a 9+ rating on the Apple App Store. The Apple metadata indicates the presence of mild cartoon or fantasy violence, whereas the Android listing does not explicitly disclose comparable content descriptors, despite the ACB Mature rating permitting moderate fantasy themes and violence (**cf.** Appendix A.1).

1.2 Vision–Language Models for CR Classification

As discussed earlier, recent advances in vision–language models offer scalable opportunities; however, in deep learning, the suitability of a model for a given task depends heavily on the alignment between the model's training distribution and the target distribution encountered at inference. When this alignment is weak or absent, performance degrades due to domain shift.

We identify that a similar domain shift occurs when applying VLMs such as GPT or Gemini [49] to the task of mobile app CR prediction. Unlike standard VLM training, which leverages conceptual captions, image–text pairs where the texts directly describe the images in the real world, app metadata consists of less indicative, loosely related modalities: screenshots and icons that represent the aesthetic identity of the app, and textual descriptions that highlight marketing intent and functionality of the app. These two modalities rarely form coherent semantic pairs. For example, a puzzle game might feature a cat image that is unrelated to its description, where players arrange fragmented cat blocks to form a complete picture, or a hand-shaped icon may indicate "Raise hand" in a video conferencing app or "Hand tool" in Photoshop.

Categories	Apple Content Rating Descriptor	4+	9+	13+	16+	18+
In-App Controlls	Parental Controls	✓	✓	✓	✗	✗
	Age Assurance	✓	✓	✓	✗	✗
Capabilities	Unrestricted Web Access	✗	✗	✗	✓	✓
	User-Generated Content	✓	✓	✓	✓	✓
	Messaging and Chat	✓	✓	✓	✓	✓
	Advertising	✓	✓	✓	✓	✓
Mature Themes	Profanity or Crude Humour	✗	Inf	F	F	F
	Horror / Fear Themes	✗	Inf	F	F	F
	Alcohol, Tobacco, or Drug Use or References	✗	✗	Inf	Inf	F
Medical or Wellness	Medical or Treatment Information	✗	✗	Inf	F	F
	Health or Wellness Topics	✗	✓	✓	✓	✓
Sexuality or Nudity	Mature or Suggestive Themes	✗	Inf	Inf	F	F
	Sexual Content or Nudity	✗	✗	Inf	Inf	F
	Graphic Sexual Content and Nudity	✗	✗	✗	✗	F
Violence	Cartoon or Fantasy Violence	✗	Inf	F	F	F
	Guns or Other Weapons	✗	Inf	F	F	F
	Realistic Violence	✗	✗	Inf	Inf	F
	Prolonged Graphic/ Sadistic Realistic Violence	✗	✗	✗	✗	F
Chance-Based Activities	Simulated Gambling	✗	✗	Inf	Inf	F
	Contests	Inf	Inf	F	F	F
	Loot Boxes	✗	✓	✓	✓	✓

Table 1.1: Apple App Store age rating descriptors and their allowed presence or frequency across age categories. Entries indicate whether a feature is permitted (✓) or allowed at an infrequent (Inf) or frequent (F) level for a given age rating.

In addition, while modern VLMs excel at identifying objects and visual entities, CR prediction does not solely depend on object-centric cues. It depends on the interplay between content (e.g., presence of weapons) and style (e.g., cartoonish vs. realistic depiction). Children's apps commonly use bright, rounded, exaggerated visual styles, whereas adult-targeted content may depict similar objects with heightened realism, darker palettes, or graphic detail. Generative models, however, are typically biased toward object recognition and exhibit limited sensitivity to stylistic properties such as colour tone, texture, or emotional ambience. This creates a style-content disparity that further reduces prediction reliability.

Taken together, the modality misalignment and style–content disparity highlight why current generative models are fundamentally unsuited for reliable CR prediction. Addressing these limitations would require not only new training data regimes, but also models that explicitly encode rating-specific features, contextual cues, content and style feature alignments.

1.3 Scope of the Thesis

Ensuring mobile app safety and regulatory compliance requires a deep understanding of how apps evolve, how their content is represented, and how automated systems can verify the appropriateness of the app content. Yet, existing app ecosystems exhibit substantial compliance

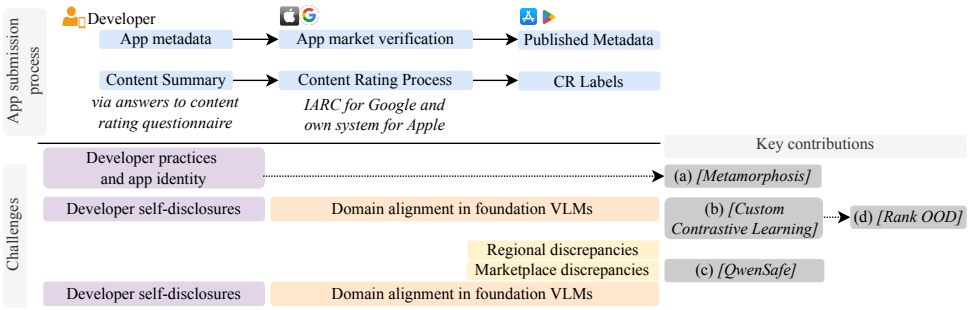


Figure 1.4: Scope of the thesis highlighting the key contributions which address the challenges in each stage of app submission process.

gaps: 1) developers can transform their apps in ways that obscure their identity and purpose, 2) content rating systems are vulnerable to self-reporting biases and inconsistencies, and 3) the nuanced descriptors remain difficult to identify reliably through automated means. Addressing these challenges requires a multi-faceted investigation that spans ecosystem-level analysis, multimodal representation learning, and fine-grained generative reasoning. Figure 1.4 shows how we address the key challenges in each chapter.

From the perspective of ecosystem integrity, we first examine whether mobile applications retain stable identities throughout their lifecycles or whether they undergo dramatic changes that escape traditional monitoring mechanisms. To this end, we introduce a novel phenomenon called app metamorphosis (cf. Figure 1.4 (a)), where apps significantly alter their purpose, presentation, or market positioning across time. Using two large-scale snapshots of the Google Play Store taken five years apart, we develop a multi-modal search methodology that combines icon similarity, textual semantics, and developer metadata to uncover hidden transformations such as re-branding, re-purposing, re-birthing, and more. This allows us to characterise the prevalence, motivations, and consequences of metamorphosis, shedding light on potential security and privacy risks that remain invisible under conventional app lifecycle assumptions.

From the perspective of content rating robustness, we investigate whether current content rating mechanisms reliably reflect the actual content presented to users (cf. Figure 1.4 (b)). The Google Play Store largely relies on developer self-disclosures obtained via questionnaires, a practice that can incentivise under-reporting of content ratings to reach broader user audiences. To address this, we propose a vision–language approach that leverages a contrastive learning technique, effectively learn representations across different modalities of metadata (app description and screenshots), obtained via textual and content/style visual encoders and particularly utilising ListMLE [56] based ordinal classifier for the prediction of ACB content rating labels. Our findings highlight large-scale discrepancies between predicted and developer-assigned ratings, revealing numerous violations. This approach not only reveals widespread content malpractices and disguises, but also highlights how certain developers strategically under-report mature themes to broaden their market reach or bypass policy constraints.

Building upon this foundation, our third contribution advances content compliance detection to a fine-grained, descriptor-aware level. Rather than predicting a single content rating, we focus on the availability of content rating descriptors (CRDs), such as cartoon or fantasy violence, mature themes, gambling, or alcohol references, which often diverge sharply across platforms (cf. Figure 1.4 (c)). To enable more interpretable analysis, we develop QwenSafe, a preference-aligned multimodal generative model designed to identify the presence of Apple-defined CRDs directly from app creatives and screenshots. Utilising a metadata2CRD pipeline, we construct descriptor-aligned supervision that enables the model to detect not only declared descriptors but also undefined, hidden, or unreported CRDs. This allows for a more transparent and evidence-driven compliance assessment and reveals subtle forms of policy evasion that remain invisible when evaluating only high-level ratings.

Collectively, these three strands: 1) ecosystem-level metamorphosis detection, 2) contrastive multimodal content rating prediction, and 3) content rating descriptor level reasoning, provide a unified investigation of the reliability, transparency, and detectability of mobile app content compliance.

Motivated by the effectiveness of our ranking-based approach for ordinal content rating classification, we expand the ListMLE loss to model the inherent ranking order among standard image classifiers such as ResNet18 [57]. We then show that deviations from the learned ranking order serve as a powerful signal for OOD detection. Our proposed method, RankOOD (cf. Figure 1.4 (d)), provides a unified framework by analysing class-wise rank order patterns directly from model outputs and without introducing auxiliary outliers during training. In Table 1.2, we summarise the key contributions of this thesis.

1.3.1 Structure of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 reviews related work across mobile app ecosystems, content rating systems, and multimodal learning, highlighting current limitations in app compliance monitoring and robustness. Chapter 3 presents our analysis of Android app metamorphosis, introducing the multimodal similarity framework used to identify large-scale app transformations and discussing their ecosystem, privacy, and security implications. Chapter 4 investigates content rating compliance using multimodal inputs, describing our contrastive learning approach for rating prediction and demonstrating how it exposes rating inconsistencies, malpractices, and developer disguises. Chapter 5 extends content compliance analysis to a fine-grained level through descriptor-aware modelling, detailing the QwenSafe framework for CRD detection and its ability to uncover undefined or unreported descriptors. Chapter 6 explores ordinal classification loss, ListMLE as a robustness signal, introducing RankOOD, a class ranking framework for detecting out-of-distribution data via violations of

Chapter	Key Contributions
Chapter 3 (Mobile App Metamorphosis)	• Novel multi-modal app similarity search framework, by jointly leveraging five different modalities via majority voting. • First large-scale measurement of app metamorphosis, analysing two Google Play Store snapshots five years apart to identify and categorise Android app transformations such as re-branding, re-purposing, and re-birth on top-10k apps. • Introduction of a quantitative success score metric that represents app growth against overall Android ecosystem growth, enabling empirical comparison of the impact of different metamorphosis strategies.
Chapter 4 (Custom Contrastive Learning)	• Propose a supervised contrastive vision–language framework that learns joint embeddings of app creatives (app icon and screenshots) and descriptions to capture age-relevant semantic cues. • Propose an ordinal content rating classification that integrates ListMLE with cross-entropy loss using learned visual and textual embeddings. • A large-scale empirical analysis of content rating violations, uncovering malpractices and disguises in the Google Play Store.
Chapter 5 (QwenSafe)	• Propose a novel metadata2CRD data construction pipeline that synthesises descriptor-aligned multimodal supervision by jointly leveraging app metadata, screenshots, and formal content rating definitions. • Propose an offline model adaptation framework combining supervised fine-tuning (SFT) with mistake-driven Direct Preference Optimization (DPO), aligning a vision–language model with descriptor-specific detail and explicitly correcting prediction errors. • Uncovers a substantial fraction of undisclosed content rating descriptors in both the Apple App Store and Google Play, revealing widespread content non-compliance
Chapter 6 (RankOOD)	• Introduce RankOOD, a novel class ranking–aware training framework, exploits the implicit class ranking structure learned during standard cross-entropy pre-training and models it explicitly using a ListMLE objective to improve OOD detection. • The method introduces a rank-based OOD score that quantifies misalignments between predicted and canonical class rankings, enabling consistent and state-of-the-art OOD detection performance by outperforming prior rank-based approaches across near- and far-OOD benchmarks.

Table 1.2: Key contributions of this thesis

learned ordinal structures. Chapter 7 concludes the thesis by summarising key findings and outlining future research directions in mobile app transparency, safety, and multimodal compliance analysis.

Chapter 2

Related Work

With continuous technological advancements, a diverse range of information can now be readily distributed to users through mobile apps. While such apps enhance convenience, they also introduce significant societal challenges, including privacy violations and the creation and distribution of unlawful or inappropriate content. Inappropriate content particularly affects children's development, while privacy violations pose serious risks to individual rights.

Regulators worldwide are increasingly focusing on preventing inappropriate content in online services, particularly content that may harm children or vulnerable users. Several legal frameworks address online safety and age-appropriate access, including U.S. regulations such as the Children's Online Privacy Protection Act (COPPA) [58], the Children's Internet Protection Act (CIPA) [59], and the Communications Decency Act (CDA) [60], which aim to restrict children's exposure to harmful content. Content rating systems for media, such as those issued by the MPAA [61] and ESRB [41], further provide standardised guidance for videos, movies, and games. Despite these efforts, the mobile application ecosystem still lacks a unified and explicit regulatory framework for app-level content rating and moderation, leaving a gap in mobile app content standards.

Many studies have examined security and privacy risks in smartphone platforms, particularly whether mobile app behaviour, such as data collection and sharing, aligns with privacy policy statements on online advertising and tracking [62, 63, 64, 65]. Other work has analysed user reviews [66, 67], cookie consent mechanisms [68], and in-app advertising content [69, 70]. In contrast, relatively few studies have focused on mobile app content/age ratings [71, 72, 53], despite growing ethical and legal concerns surrounding inappropriate content and content suitability in mobile applications (a.k.a apps). As discussed in Section 1.1, app markets and regulatory bodies have implemented rules, policies, and processes to assign content ratings [73, 74] to mobile applications; however, these procedures are time-consuming and prone to errors and inconsistencies. Recently, image- and text-based contrastive learning and multimodal large language models have shown promise in reducing reliance on labelled data and

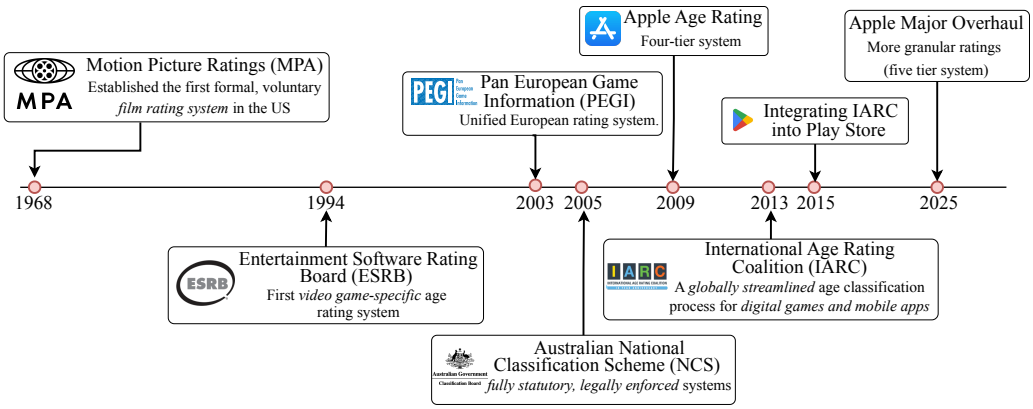


Figure 2.1: Evolution of content rating schemes.

enabling effective representation learning, understanding, and reasoning, providing a viable alternative to traditional, supervised learning based content appropriateness detection.

The remainder of the chapter is organised as follows. Section 2.1 discusses how app markets have evolved to reach the current content rating schemes. Section 2.2 reviews research on online privacy violations, including profiling, consent breaches, content rating malpractices and in-app advertisements. Section 2.3 presents self-supervised learning with image and text modalities, focusing on contrastive learning techniques and applications. Finally, Section 2.4 discusses Multimodal Large Language Models (MLLMs).

2.1 The Evolution of Content Rating Schemes

Modern rating practices originated with the introduction of the first voluntary film rating in 1968, which established the template for advisory and replaced the outdated and highly restrictive self-censorship known as the Hays Production Code. This model, using age categories and content descriptors to inform consumers, was later adopted by the video game industry two decades later. Consequently, the Entertainment Software Rating Board (ESRB) was established in 1994 in response to U.S. congressional scrutiny of violent game content. Europe followed with the Pan-European Game Information (PEGI) system in 2003, creating a unified cross-national standard grounded in co-regulatory principles. Parallel national developments, such as Germany's USK and Australia's National Classification Scheme (NCS), contributed to a heterogeneous global regulatory landscape; the Australian system, formalised under the Classification Act 2005 and expanded in 2013 to include an R18+ category, represents one of the few fully statutory and legally enforceable video game classification regimes worldwide.

After the ESRB and PEGI successfully established trusted, self-regulatory systems, any developer wishing to sell their game across major global markets had to engage with multiple different rating authorities. This often required submitting separate, detailed applications and

waiting for assessment by each body. This bottleneck became unsustainable as the digital distribution markets, such as mobile app stores and PC stores, exploded. The launch of IARC in December 2013 was a direct response to this challenge, which streamlined cross-regional ratings for mobile and online platforms. It represented a groundbreaking agreement where the world's major rating authorities (including their participating bodies ESRB, PEGI, ACB, and USK) collaborated to create a single, automated questionnaire.

Age rating systems for digital content vary substantially across jurisdictions in terms of legal status, enforcement mechanisms, and institutional design. In the United States, the ESRB operates as a fully self-regulatory and advisory system without legal enforceability [75, 76]. However, the ESRB framework is partially enforceable in certain Canadian provinces, such as Ontario, where age classifications may carry regulatory weight. European systems adopt co-regulatory models with varying degrees of legal enforceability. For example, PEGI and the German USK may be legally binding in specific jurisdictions while remaining advisory in others, and both incorporate input from non-industry stakeholders to mitigate conflicts of interest [77]. Unlike advisory systems such as the ESRB, Australia's classifications are enforceable by law [78]. Until 2013, games could not receive ratings higher than MA15+, leading to the refusal of classification for titles deemed unsuitable, and games containing gambling were often rated G or PG and widely accessible to minors [79]. After years of debate, Australia introduced an R18+ rating for games in 2013, replacing the older system where anything above MA15+ was banned [80]. Recently, ACB introduced a new minimum rating [M-Mature] for chance-based purchases (e.g. loot boxes) and R18+ for simulated gambling.

The exponential growth of the mobile app market presented a monumental challenge to the traditional age rating structure and the direct integration of the IARC system into the development dashboards. Most notably, Google Play adopted IARC ratings in March 2015, followed by other platforms like the Microsoft Store. However, fragmentation persists, particularly as major actors such as Apple continue to employ proprietary rating schemes. In 2009, Apple introduced its own rating scheme, a four-tier rating scheme. Later in 2025, Apple expanded it to a five-tier rating by introducing 13+, 16+ and 18+ and excluding 12+ and 17+ ratings from the old scheme. Fig 2.1 depicts the major evolutionary changes happened in age rating schemes.

2.2 Empirical Studies on Privacy Risks of Mobile Apps

This section summarises the privacy malpractices in mobile apps, along with corresponding detection methods. As shown in Figure 2.2, these can be classified into four categories: 1) privacy legislation violations, 2) in-app advertisement violations, 3) consent notice violations, and 4) content malpractices.

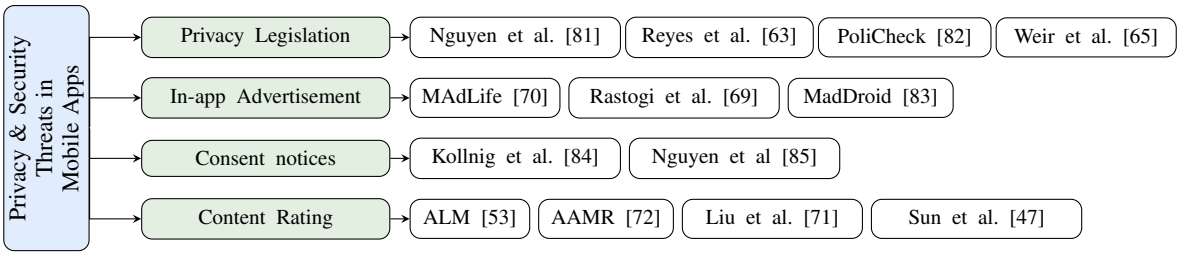


Figure 2.2: Summary of prior research on detecting mobile app privacy and security threats.

2.2.1 Detecting Non-conformity to Privacy Legislation

While web tracking and personal information sharing have been widely studied [68, 86], research on mobile applications has only recently gained attention, as mobile apps are often legally permitted to collect and share data if declared in their privacy policies. Under the California Consumer Privacy Act (CCPA) [29], app developers must provide accurate privacy notices and respond to *right to know* requests by disclosing any personal information collected, used, or shared for commercial purposes. Similarly, the GDPR requires developers to obtain explicit, *freely given, specific, informed, and unambiguous* consent before sharing personal data. Common violations include tracking without consent, profiling cookies, and not allowing the user to deny data collection [63, 68, 87, 65].

Weir et al. [65] surveyed the impact of GDPR on app developers, finding that while cosmetic changes were common, GDPR had a limited effect on improving user privacy. Reyes et al. [63] proposed a dynamic analysis framework for Android to automatically detect potential COPPA violations, observing app behaviour in real time using Google Play's 'Designed for Families' apps. Building on this, PoliCheck [82] combines dynamic analysis of app data flows with privacy policies to verify entity-sensitive flow-to-policy consistency, ensuring that mobile apps accurately disclose privacy-sensitive data flows and explicitly account for the recipients (first- vs. third-party). Zimmeck et al. [62] developed a machine learning classifier to categorise policy content using keyword search, bi-grams, and word modifiers.

2.2.2 Detecting In-app Advertisement Threats

Mobile advertisements (ads) are a major revenue source for app developers, accounting for 52% of digital advertising spending in Australia in 2021 (6.8 billion) [88]. However, the ad ecosystem is vulnerable to abuse, including unintended ad clicks used to defraud advertisers and users. In 2015, mobile ad fraud caused losses of 1.3 billion in the U.S. Research has increasingly focused on this area, including automated ad library identification [89, 90], mobile ad fraud detection [91, 92], and security and privacy analysis of ad libraries [93, 94]. Ad content also poses risks: downloading an app that is advertised at runtime by a legitimate app may contain malware or can be a malicious app, and some Google Play apps have been found to display

pornographic ads, violating content policies [95, 96].

Chen et al. [70] were the first to analyse in-app advertisement content. They classified the ads into two categories: policy non-compliant and controversial, by analysing screenshots of ad landing pages using the Google Vision API. They further introduced MAdLife, a framework for collecting in-app ad traffic on mobile platforms that is generated during an ad's lifecycle. Rastogi et al. [69] proposed a framework to analyse ad-clicking app–web interfaces, that redirect users from the app to external web pages, to detect deceptive content and identify malicious origins. Their approach includes: (1) identifying ad-triggering buttons or widgets using edge detection and contour recognition, (2) detecting malicious landing pages or URLs via VirusTotal, and (3) providing the source of malicious activity. In contrast, MadDroid [83] detects fraudulent ad loading and clicking behaviours and classifies them into five categories using a plug-in–based architecture.

2.2.3 Detecting Deceptive Practices in Consent Notices

Each time a mobile app is used, user information and online activities are transmitted to third-party entities, primarily for advertising purposes. To limit user tracking, governments have introduced regulations such as the ePrivacy Directive [97], GDPR, and CCPA. The ePrivacy Directive and Recital 30 of the GDPR require explicit user consent for sharing personal data, obligating mobile apps to inform users of the consent purpose and provide the option to decline specific uses. Despite these safeguards, researchers have identified deceptive practices that undermine user control, including pre-selecting affirmative consent, assigning consent irrespective of the user's choice, and excluding rejection options. As a result, enforcement of GDPR consent requirements remains limited.

An early study on consent notices in mobile apps was conducted by Kollnig et al. [84], who manually analysed 1,297 Google Play apps to assess the presence and quality of consent notices. Nguyen et al. [81] later performed the first large-scale analysis, focusing on three aspects: 1) common characteristics of consent notices, 2) GDPR legal justification, and 3) developer awareness of GDPR and related violations. They crawled free Android apps from Google Play Store between October 2021 and March 2022 using the AndroZoo dataset (European Economic Area location based selection). Using TF-IDF and hypernym-based feature generation, formed 44 clusters and found that only 13,082 out of 250,972 apps implemented consent notices.

2.2.4 Detection of Mobile App Content Rating Malpractices

Maturity ratings are used to identify whether there is harmful/inappropriate content in mobile apps. Notably, while selecting apps for their children and teenagers, parents place great value on app maturity ratings, which necessitate the importance of an accurate rating system. However, research on automatic maturity detection or evaluation for mobile apps is limited.

Chen et al. [70] studied the reliability of maturity ratings in Google Play and the Apple App Store, showing that iOS ratings are generally more precise due to stricter manual review processes. They proposed a framework to verify maturity ratings across platforms and introduced Automatic Label of Maturity ratings (ALM), a text-mining approach that ranks apps using descriptions and user reviews with manually selected seed lexicons. However, ALM relies on keyword matching, ignores semantic context, and requires costly manual labelling. Hu et al. [72] proposed the Automatic App Maturity Rating (AAMR) framework, which predicts ratings based on policy guidelines using app descriptions encoded with Word2Vec and TF-IDF features and classified via an SVM-based multi-label approach. Despite incorporating semantic information, AAMR relies solely on text features and requires large-scale human annotation. Liu et al. [71] developed a multimodal learning approach that uses both image and text features to predict an app's suitability for children, incorporating app icon colour distributions, screenshot text characteristics, and metadata features. Sun et al. [47] analyse content age ratings by collecting ratings from multiple authorities (ESRB, PEGI, USK, ACB, IARC) and quantifying cross-authority inconsistencies using an age-gap–based metric, complemented by manual review.

A separate line of work focuses on detecting inappropriate content in mobile apps rather than predicting app content ratings. HeCGamb [98] analyses encrypted network traffic by collecting data generated during app execution, extracting TLS handshake fingerprints and flow statistics, and using a heterogeneous graph neural network to classify apps as gambling or benign. Serra et al. [99] address the privacy-preserving detection of sensitive media (e.g., pornography and violence) in messaging apps by proposing a media moderation framework that analyses exchanged media via an external MediaClassifier service without inspecting message text. Their approach employs a multimodal pipeline that extracts visual features using InceptionV3 and audio features using AudioVGG, aggregates them into a joint embedding, later applies an SVM classifier to identify sensitive content.

Existing work focuses on privacy violations under legislation, with limited attention to consent notice exploitation and misleading in-app advertisements. Mobile app content rating analysis remains relatively unexplored. Prior studies target child-directed apps or content rating prediction via supervised learning or focus on a single type of inappropriate content, requiring costly labelled data and often relying on a single modality. The next section reviews recent multimodal self-supervised learning approaches and examines how large-scale unlabelled multimodal data is adapted to generate content rating of apps and the unique challenges that arise in these settings.

2.3 Overview of Multimodal Self-Supervised Learning

Supervised learning depends on large-scale human-annotated datasets, which are expensive and time-consuming to collect. However, most real-world data is unstructured, making label acquisition difficult and often impractical, particularly in specialised domains such as medical imaging that require expert knowledge. To overcome the limitations of supervised learning, several alternative paradigms have been proposed, including self-supervised learning (SSL). Although the term self-supervised learning (SSL) was introduced later, related ideas have existed for decades. SSL methods learn discriminative representations from large-scale unlabelled data without requiring human annotations. However, SSL received limited attention until its success in natural language processing. The introduction of Word2Vec by Mikolov et al. [100] demonstrated the effectiveness of self-supervised objectives for learning meaningful representations, paving the way for large-scale language models such as BERT [101], RoBERTa [102], and XLM-R [103]. More recently, SSL has gained significant interest in computer vision and later in multimodal learning due to its ability to learn representations directly from data without manual annotation. In this section, we provide an overview of multimodal SSL.

SSL follows a two-stage pipeline. In the self-supervised pre-training stage, a predefined pretext task is constructed, where pseudo-labels are automatically generated from intrinsic properties of the input data. The model is trained to solve this task without human annotation. The pre-trained model is then transferred to downstream tasks. Pretext tasks generally share two characteristics: (1) deep networks are trained to learn representations useful for solving the task, and (2) supervision is obtained intrinsically from the data. However, multimodal SSL differs from unimodal approaches by requiring objectives that account for the characteristics of each modality. Common pretext tasks in multimodal SSL can be grouped into three categories: clustering, instance discrimination, and masked prediction, highlighting extensions of classic unimodal objectives to handle multimodal inputs.

Clustering: Clustering-based approaches assume that end-to-end training induces representations in which samples are grouped according to semantically meaningful features. These methods iteratively predict cluster assignments of encoded representations and use them as a pseudo-label, a supervisory signal to update features. Multimodal clustering leverages pseudo-labels from one modality to supervise others, capturing cross-modal structure and improving downstream tasks like retrieval. Cluster assignments can be derived from jointly fused or modality-specific representations. Representative methods include Cross-Modal Deep Clustering (XDC) [104], which supervises one modality using cluster assignments of the other modality. SeLaVi [105], which extends unimodal SeLa [106] via an optimal transport problem to learn audio-visual view invariance; and DMC [107], which co-clusters image and audio features and uses cross-modal similarity as supervision. Unlike unimodal clustering, multimodal clustering

can allow different modalities to have distinct cluster assignments, increasing diversity when paired modalities are imperfectly aligned. However, this flexibility is dataset-dependent and difficult to tune a priori. Further, clustering is sensitive to initialisation, clustering algorithm and the number of clusters must be carefully chosen to avoid over- or under-fitting.

Masked Prediction: masked prediction can be formulated as auto-encoding, as in BERT [101], or auto-regressive, as in GPT series [108, 48]. Auto-encoding approaches pre-train models by masking input elements and predicting the missing information. Originally introduced as masked language modeling (MLM) for NLP, this strategy is now widely applied in multimodal learning. In cross-modal settings, the model predicts missing information conditioned on other modalities. For example, the model can use an image as context to predict missing text or use text to predict missing visual content. SelfDoc [109] randomly masks language or vision features for document understanding, while VideoBERT [110] converts video and text into discrete tokens, enabling pre-training with intra-modal mask-completion on either video- or text-only corpora using the same language model. Auto-regressive pre-training, introduced by PixelCNN and GPT [108], predicts masked tokens sequentially from left to right. SimVLM [111] improves this with a PrefixLM objective, applying bidirectional attention to the prefix and auto-regressive factorisation only to remaining tokens, efficiently leveraging cross-modal information while reducing training complexity.

Masked prediction has proven effective in unimodal settings, such as language (BERT [112], GPT [108]) and vision (MAE [113]). By tokenizing inputs, the same objective can be applied across modalities, and mixing different modality tokens enhances cross-modal interactions. However, a key limitation is the added computational cost from the decoders required to reconstruct masked inputs.

Instance Discrimination: Instance discrimination aims to determine whether samples from two input modalities correspond to the same instance (i.e., are paired). By enforcing this objective, the method aligns the representation spaces of paired modalities while pushing representations of mismatched instance pairs further apart. Depending on the sampling strategy, instance discrimination objectives can be broadly categorized into contrastive learning (CL) and matching prediction. Visual CL methods treat views of the same instance as positive pairs and views from different instances as negative pairs, encouraging closeness for positives and separation for negatives in the latent space. Notable examples in the visual domain include MoCo v1 [114], MoCo v2 [115], and SimCLR [116].

In multimodal CL, corresponding modalities (e.g., image and caption) form positive pairs. Contrastive Multiview Coding (CMC) [117] was an early multimodal approach, maximizing mutual information between different modalities of the same scene while separating unmatched samples. This principle of cross-modal instance discrimination has since been widely extended. MultiModal Versatile (MMV) networks [118] maximize mutual information among

temporally co-occurring vision, audio, and text pairs. Similarly, Video-Audio-Text Transformer (VATT) [119] employs a modality-agnostic, single-backbone transformer with shared weights across the three modalities using a contrastive objective.

Prior research has shown that CL scales effectively to large models and datasets. CLIP [50] achieves strong zero-shot performance by predicting pairings on 400M image-text pairs. This approach has been extended to AudioCLIP [120], VideoCLIP [121], and pointCLIP [122]. ALIGN [51] shows that pre-training on noisy image text pairs still yields strong performance. To improve efficiency for large-scale training, CLIPPO [123] unifies CLIP modalities using a pixel-based approach, and FLIP [124] masks large image patches, achieving performance comparable or superior to CLIP. In addition to inter-modal alignment, intra-modality contrastive learning provides complementary supervision. SLIP [125] combines intra-modal and cross-modal contrastive losses, improving performance on the image-text task.

Matching prediction, also known as alignment prediction, is widely used in large-scale multimodal learning. It classifies whether a pair of samples from two modalities forms a positive or negative pair(e.g., if a piece of text corresponds to an image's caption). Unlike contrastive learning, which compares a positive pair against all negatives in a mini-batch, matching prediction labels individual tuples as positive or negative. Image-Text Matching (ITM), first introduced by UNITER [126], feeds global cross-modal representations to a binary classifier to predict pair alignment. ITM has been adopted in models such as BLIP [127] and FLAVA [128].

Instance discrimination is effective but often requires large batch sizes to ensure sufficient negative samples, making it memory-intensive [129]. Cross-modal contrastive models compute dot products between embeddings from modality-specific encoders, which is simple but limits cross-modal interactions. In contrast, matching prediction operates on joint representations, enabling richer interactions. Combining both objectives can provide complementary benefits.

2.3.1 Applications of Multimodal Self-Supervised Learning

Self-supervised multimodal learning methods have been widely applied across real-world domains, including state representation learning, healthcare, remote sensing, autonomous driving, and machine translation. In healthcare, clinical decision-making relies heavily on multimodal data, however, the need for high-quality annotations from domain experts makes supervised learning difficult to scale, making self-supervised learning an effective alternative. In medical imaging, contrastive methods such as ConVIRT [130], and GLoRIA [131] leverage image–report pairs to improve diagnostic performance. MEDCLIP [132] decouples image and text encoders to leverage readily available unpaired medical images and reports, while ContIG [133] applies a contrastive objective to align retinal imaging and genetic modalities. In remote sensing, complementary data from multiple sensors—such as hyperspectral, multispectral, LiDAR, and SAR, enable richer earth observation. Hyper-spectral images capture land-cover

Domain	Key Contribution
Healthcare	ConVIRT [130]: medical image representation using CL on image–report pairs. GLoRIA [131]: attention-based global–local CL framework. MEDCLIP [132]: align unpaired medical images and reports via a decoupled CL model. ContIG [133]: aligns retinal imaging and genetic data via contrastive objectives.
Remote Sensing	Chen et al. [134]: aligns SAR and optical imagery using contrastive learning. Montanaro et al. [135]: applies masked prediction to fuse multispectral and SAR data. Saha et al. [136]: combines clustering and CL for optical–SAR change detection.
Autonomous Driving	GtN [137]: CLIP-based vision–language navigation for vehicle movement control. CLIP2Scene [138]: learns pixel–point–text correspondences for 3D scene understanding.
Mobile App UI Understanding	Lexi [139]: visually grounded textual representations of UI screens using SSL
Recommender Systems for Mobile Apps	SMAR [140]: improves mobile app recommendation via item representation learning. Yao et al. [141]: addresses long-tail and sparse data using SSL on app metadata.

Table 2.1: Applications of multimodal self-supervised learning across domains.

categories via spectral signatures, while SAR provides dielectric properties. SSL provides a way for integrating these heterogeneous remote sensing data, such as aligning SAR and optical images using contrastive losses [134], while masked prediction has been applied to fuse multispectral and SAR modalities [135]. Using bi-temporal scenes from heterogeneous sensors, a clustering method combined with contrastive learning is employed for detecting changes in optical–SAR images [136]. In autonomous driving, GtN [137] proposes a vision language navigation tool to control vehicle movements with the help of CLIP. CLIP2Scene [138] investigates the use of CLIP for 3D scene understanding in autonomous driving. By bridging pixel–text and pixel–point correspondences, it constructs point–text and pixel–point–text pairs to train contrastive representations. Lexi [139] proposes a vision language SSL model to learn visually-grounded textual representations of web and mobile app UI screens and their components. SMAR [140] and Yao et al. [141] use SSL as an auxiliary task to improve item representations, particularly for long-tail and sparse data, in mobile app recommendation. However, their approach only relies on text-based metadata from app landing pages, including app ID, developer name, and title. Table 2.1 summarises the adaptation of SSL in different domains.

While multimodal self-supervised learning does not rely on large-scale labelled data and aligns heterogeneous modalities across different applications, the mobile app domain, particularly content rating, remains underexplored. Adapting multimodal SSL to mobile apps is non-trivial, as existing models are trained on conceptual captions grounded in real-world images. In contrast, mobile app visuals lack well-aligned captions, and app descriptions neither reflect content appropriateness nor align reliably with screenshots.

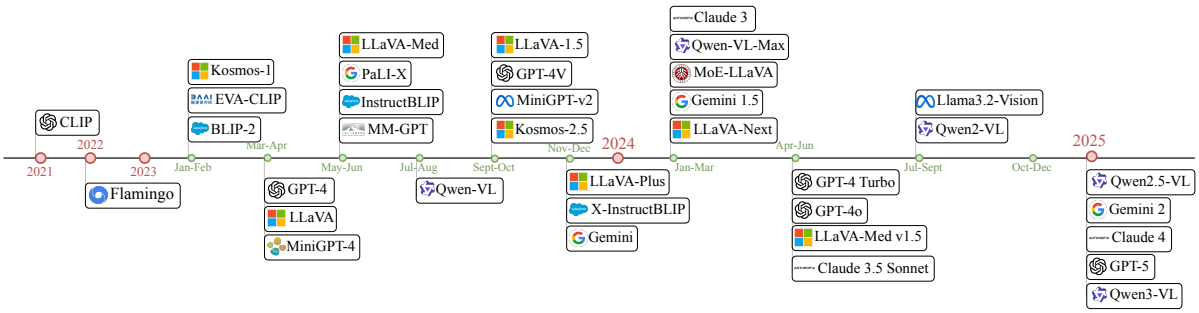


Figure 2.3: Timeline of representative text–image multimodal large language models (MLLMs) from 2021 to 2025.

2.4 Multimodal Large Language Models

Recent progress in multimodal (MM) pre-training has significantly improved performance across a wide range of downstream tasks. Despite these advances, the development of effective multimodal large language models (MLLMs) faces several fundamental challenges, particularly in visual representation quality and vision–language alignment. Most MLLMs are pre-trained using self-supervised learning, where naturally paired multimodal data (e.g., image–text pairs) provide the supervision signal. Figure 2.3 shows the timeline of representative MLLMs.

MLLM architecture and training: MLLMs integrate pre-trained models from different modalities and enable collaborative inference through effective cross-modal alignment. Architecturally, an MLLM consists of three components: a pre-trained modality encoder (e.g., vision), a pre-trained LLM serving as the decoder, and a learnable modality interface that bridges the two. The LLM backbone is typically kept architecturally unchanged, with optional lightweight adaptations such as LoRA. Widely used MLLMs generally follow a three-stage training pipeline: pre-training, instruction tuning, and alignment tuning. Pre-training focuses on aligning modalities and learning multimodal representations using large-scale paired data (e.g., image–caption pairs), often by freezing both the encoder and LLM and training only the interface modules. Instruction tuning further adapts the model to multimodal, instruction-following tasks by jointly training the interface and LLM on curated instruction datasets, improving zero-shot and few-shot generalisation. Finally, alignment tuning refines model outputs to better match human preferences, commonly using reinforcement learning with human feedback (RLHF) or direct preference optimisation (DPO).

Visual Encoders: Recent work has focused on improving visual encoders to better capture fine-grained visual information. This line of research has progressed along two main directions: 1) enhancing the representation capabilities of individual encoders and 2) combining multiple encoders in a complementary manner. Early encoders such as CLIP [50] provide strong high-level semantic representations, however lack detailed geometric information. Subsequent models, including SigLip [142] and Eva-CLIP [143], improve semantic alignment and fine-grained

recognition through optimised training objectives and data construction, while the DINO series [144] captures pixel-level geometry and texture via self-supervised learning. More recent approaches progress towards multi-encoder approaches, for example, combining CLIP with DINOv2 [145] to leverage both semantic and geometric strengths. Methods such as LLaVA-HR [146] and Mini-Gemini [147] further incorporate high-resolution visual features through hybrid-resolution adapters and localised cross-attention mechanisms, respectively.

Pre-trained LLMs: Rather than training from scratch, it is more efficient to initialize with a pre-trained LLM. Extensive pre-training on large-scale web corpora embeds LLMs with rich world knowledge, strong generalization and reasoning capabilities. Most commonly used LLMs fall under the decoder-only category. Early models such as BLIP-2 and InstructBLIP employed Flan-T5 series. The LLaMA series [148] and the Vicuna family [149] are widely studied open-source LLMs, but their pre-training primarily on English corpora limits multilingual capabilities, particularly for Chinese. In contrast, Qwen [150] is a bilingual LLM supporting both Chinese and English. The research indicates that scaling LLMs yields consistent performance gains for MLLMs but incurs substantial computational cost. In contrast, mixture-of-experts (MoE) architectures have attracted attention for their sparsity, and empirical results from MoE-LLaVA [151] show that MoE models outperform dense counterparts across most benchmarks.

Vision–language alignment: Effective projection layers must not only increase information throughput but also enable dynamic, task-aware feature selection and structured visual-to-textual feature transformation. Existing alignment strategies can be broadly categorized into three types: 1) deep fusion using standard cross attention layers, 2) deep fusion via custom layers and 3) early fusion. Flamingo [152] directs resampled tokens into standard cross-attention layers of the LLM, which augment language representations with visual information. In contrast, mPLUG-Owl2 employs custom cross-attention layers that are fused into the internal layers of the pre-trained LLM model [153]. mPLUG-Owl2 learns a modality adaptive module, a custom cross attention layer, which employs a common Q for separate K and V in each modality. Instead of inserting custom cross-attention into the LLM, MoE-LLaVA modifies each decoder layer of LLaVA by adding LoRA weights. Although these deep fusion approaches enable fine-grained modality interaction, they require substantial training data and computational resources. In contrast, early fusion methods directly integrate the interface output at the LLM input, avoiding modifications to internal layers. Common interface designs include linear-layer/MLP [52, 154], Q-former [155], attention-pooling layer, custom learnable layer [156] or perceiver resampler [157]. These approaches are more scalable due to its modular nature, do not require the internal layers of the LLM and the number of trainable parameters is lower compared to the other models.

2.4.1 Applications of Multimodal Large Language Models

Beyond general-purpose assistants, some studies target domain-specific scenarios that account for practical constraints, while others extend MLLMs to downstream tasks that require specialised expertise. A common trend is to adapt MLLMs to real-world, task-specific scenarios. For example, previous work develops interactive agents for real world scenarios, such as GUI assistants (e.g., CogAgent [158]), as well as embodied agents [159] capable of reasoning, navigation, and manipulation. These systems typically emphasise task planning and step-wise execution based on user instructions. Another line of research augments MLLMs with domain-specific capabilities. In document understanding, models such as mPLUG-DocOwl [160] leverage diverse document-level supervision to improve OCR-free understanding. MLLMs have also been extended to traditional vision tasks, including visual grounding, by unifying input–output formats under a language modeling objective and reformulating the grounding task as conditioned box coordinate prediction [161, 162]. Additionally, MLLMs have been adapted to medical domains by incorporating specialised knowledge, as demonstrated by LLaVA-Med [52].

In the mobile application domain, MobileAgent [163] and AppAgent [164] leverage existing MLLMs such as GPT-4V to build UI agents, primarily relying on prompt engineering and external modules or APIs to access UI-layer information. Recently, lightweight MLLMs have gained attention for efficient deployment in resource-constrained settings [165]. These models are designed to minimize computational and memory overhead while largely preserving performance, making them suitable for deployment on mobile devices. From a modeling perspective, various strategies have been explored to enable efficient training and inference. For example, MobileVLM [166] develops compact MLLM variants tailored for limited-resource environments, employing techniques such as smaller LLM backbones and quantization to accelerate computation. Similarly, MiniCPM-V [167] targets efficient end-device inference and adopts a Q-Former [168] to reduce the number of visual tokens per image patch.

A separate line of work incorporated MLLMs to detect content safety in social media memes. The Hateful Memes challenge [169] established a rigorous multimodal benchmark by contrasting subtle hateful content with benign samples, thereby discouraging unimodal approaches. Pro-Cap [170] leverages a frozen CLIP-based VLM to identify hateful meme content, while MOMENTA [171] employs a multimodal reasoning framework for harmful meme detection. UGCG-GUARD [172] is proposed to detect image-based unsafe user-generated content. It builds on a conditional prompting strategy with chain-of-thought reasoning and leverages InstructBLIP to enable zero-shot adaptation.

Although multimodal large language models, particularly vision–text models, have advanced rapidly and have been applied to content safety tasks such as hate speech detection on social media, none have focused on mobile app content rating, which requires not only detecting inappropriate content but also incorporating age restrictions, impact, and intensity.

Chapter 3

Detecting and Characterising Mobile App Metamorphosis in Google Play Store

As outlined in the scope of this thesis (**cf.** Section 1.3), this chapter[1] investigates the evolution of Android apps over the years. The primary research question addressed in this work is how developers sustain application popularity over time and what long-term evolutionary patterns emerge in mobile app ecosystems.

3.1 Introduction

The mobile app industry is highly competitive, with over 2.5 million apps on the Google Play Store [173] and 1.5 million apps on the Apple App Store [174], it can be difficult for apps to stand out from the crowd. To be successful, app developers need to create innovative and user-friendly apps, update existing apps, and respond to new operating system features and market demands.

While the typical life cycle of an app involves releasing it to the app market and periodically updating it to add new features or fix bugs and security issues, some apps deviate from this pattern of incremental updates. Instead, they undergo dramatic changes in their use cases or market positioning. The methods to match apps between two different time spans and observe such dramatic transformations remain largely unexplored. We define this relatively unknown phenomenon as *"app metamorphosis"* and there are multiple reasons and implications of it based on our findings.

For instance, sometimes developers may use the *re-branding* strategy to uplift a struggling app. This re-branding will involve changing the app's name, logo, or overall look and feel. The aim is to improve the app's image or to make it more relevant to a new target audience.

[1]This chapter includes the work in: **D. Denipitiyage**, B. Silva, K. Gunathilaka, S. Seneviratne, A. Mahanti, A. Seneviratne, S. Chawla, "Detecting and Characterising Mobile App Metamorphosis in Google Play Store", accepted in IEEE Transactions on Mobile Computing.

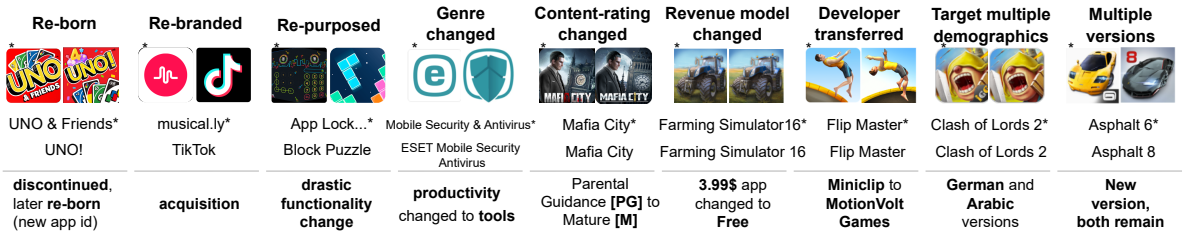


Figure 3.1: Identified app metamorphosis categories and a notable example pair for each category. (*: 2018 app)

Re-branding can also happen after a merger or an acquisition, as in the case of the popular short video-sharing social networking app *TikTok*, which used to be called *Musical.ly* with a slightly different purpose of sharing short lip-synced videos [175].

Similarly, some developers looking to extend the useful life of their successful apps employ the *re-purposing* strategy. Re-purposing an app means changing its core functionality or purpose. This enables the developers to change their app's core functionality while retaining its app audience, download numbers, ratings, etc. Equally, some developers may re-purpose their apps when their core functionalities become redundant due to them being added to the operating system as standard features. For example, due to hardware advances in smartphone cameras, some camera apps such as *Noah Camera* became photo editors.

Developers can also attempt to revive apps by *re-birthing* them. That means re-introducing a previously discontinued or an about-to-change app under a new identity (i.e., a new app ID), with minimum or no functionality change compared to the original one. This is a major undertaking, but it can be a successful way to create a new and improved app that is more likely to succeed. As an example, once *UNO & Friends* was discontinued by Ubisoft, it was revived back by a different developer *Mattel163* as *UNO!* and became successful by reaching a wider audience. Another potentially adverse reason for *re-birthing* is to overcome bans imposed by Google on developers. In Figure 3.1 we provide a summary of these metamorphosis scenarios we identify.

To date, motivations for *app metamorphosis* are not fully understood as there have been no systematic studies that focus on this aspect of the mobile app ecosystem. An investigation of *app metamorphosis* that focuses on the characteristics and prevalence of *app metamorphosis* will provide further insights into how the mobile app ecosystem is evolving. Moreover, such a study will help app market operators to identify different forms of *app metamorphosis* and their impact on the apps and enable developers to understand the implications of their actions. It is also important to understand how successful or unsuccessful the apps that underwent metamorphosis are. This will provide insights to the app developers to make decisions on app updates and monetisation strategies. Finally, there can be privacy and security implications because of these app transformations.

To this end, in this chapter, we compare two snapshots of the Google Play Store that are five years apart and propose a search methodology to identify similar apps across datasets that will help identify apps that possibly went through a "metamorphosis" phase. Next, we provide a taxonomy and a set of rules to categorise various cases of "metamorphosis" identify and discuss reasons behind them. More specifically, we make the following contributions.

- We propose a multi-modal search methodology for identifying similar apps across datasets, incorporating dimensions such as app icon similarity, app description similarity, app name, and developer name similarity. Usually, app re-identification is straightforward, and you can use the *app ID* to do that. However, in this case, we want something beyond that because part of our objective is to identify similar apps with different *app IDs* between datasets.
- We apply our search methodology to analyse two snapshots of the Google Play Store five years apart, focusing primarily on the top 10,000 apps in 2018, and identify apps that have possibly gone through "metamorphosis" between 2018-2023 and provide a taxonomy of various cases of "metamorphosis" and build a rule set to identify apps belonging to those cases.
- We introduce a "success score" to quantitatively characterise how successful an app progressing from 2018 to 2023 is, compared to the natural growth in Android eco-system. Based on the overall results, we show that app *re-branding* has the highest possibility of becoming successful ($\sim 11.3\%$ better) followed up by *re-purposing* ($\sim 4.3\%$ better) while *re-birthing* is the most challenging.
- We further discuss how metamorphosis can ascertain privacy and security risks, for example, an adverse developer could introduce a counterfeit app, disguising as a legitimate re-birth for discontinuing apps.

The rest of the chapter is organised as follows. In Section 3.2, we present the related work, and in Section 3.3 we present our datasets. Section 3.4 presents our multi-modal app similarity search algorithm and Section 3.5 presents its performance in controlled experiments. Section 3.6 presents the results and findings of using our search methods to identify apps demonstrating "metamorphosis" between 2018 to 2023. Section 3.7 discusses the privacy and security implications of our findings. Finally, Section 3.8 concludes the chapter.

3.2 Related Work

Various empirical studies have been conducted on Google Play Store and other app stores focusing on different aspects such as; general statistics related to apps [176, 177], advertising and analytics library behaviours [178, 179], characterising and detecting malware and malpractices [180, 181, 182, 183, 184, 185], and privacy and security analysis of one specific type of apps [186, 81]. However, only a limited amount of work focused on studying the evolution of

apps over a longer time period.

App life cycle in Google Play - The usual life cycle of an app in Google Play Store involves the developer publishing the app by selecting some of the metadata such as category, content rating, and self-reporting data collection and sharing practises [187, 81]. Once it is published, the app will be updated periodically to release new features or fix bugs. The vast majority of apps will continue to be like this; however, some apps may get discontinued (i.e., no further updates), removed from Google Play Store (i.e., either developer decides to remove the app, or Google removes the app or ban the developer for various reasons [18]) or go through "metamorphosis" - which is the focus of this chapter.

To date, the app life cycle has been characterised predominantly from an app update point of view. For instance, Pothuraju et al. [188] claim that nearly 76% of apps did not get any update in Play Store within their monitoring dataset for a period of approximately six months while a minority got nearly hundreds of updates which may point to newly released apps that could require many bug fixes. Though these updates are supposed to fix bugs or provide better security, Moller et al. [189] noted that even after a week of an update, app users still tend to use the older version. Vincent et at. [190] used PlayDrone [176] to extract permission usage of apps and observed that popular apps request additional dangerous permissions with subsequent updates.

Another body of work checked how some individual app libraries, such as advertising and analytic libraries, are being updated [191, 192]. However, in all these cases, the apps do not go through "metamorphosis", and app developers remain the same. We also note that the app life cycle we consider in this chapter is different from the app life cycle considered by some works [193, 194], which focus on the app usage point of view, such as when a user installs an app, and abandon it after using it for a period of time.

Disguising (or misleading) app users - Beyond malware, multiple works have explored various disguises and malpractices happening in Google Play Store. One of the common ways to distribute malware in the mobile ecosystem is through repackaging of legitimate apps. This process includes app downloading, de-compiling, manipulating, recompiling an application, and publishing it again in an app store. These repackaged apps are often distributed through third-party or unofficial app markets but can also appear on official app stores like Google Play under new app IDs. Several works looked into detecting repackaged apps by analysing bytecode similarity [195, 196, 197, 198, 199], app metadata such as APIs, permissions, and description [200, 201, 202], or runtime behavior monitoring [203]. While these works revealed privacy issues attached to mobile apps, our approach contributes differently, to a novel perspective of app life cycle changes. However, such work is related to ours because when detecting "app metamorphosis", we can incur some of these cases as false positives. Karunanayake et

al. [22] proposed a multi-modal neural framework to identify counterfeit apps that impersonate popular apps. In contrast, Viennot et al. [176] used clusters of similar apps, developer names, and certificate information to identify potential legitimate re-branding. As we present later in the results, some of the "app metamorphosis" cases we identify will have some overlaps with counterfeiting. Other forms of malpractices that have been investigated include: miscategorisation [204] and spamming [18, 17].

To the best of our knowledge, "app metamorphosis" hasn't been studied or characterised to date, and ours is the first study that investigates this phenomenon by comparing two large snapshots in Google Play Store that are five years apart.

3.3 Datasets

We use two main datasets in our work. These are two snapshots of the Google Play Store, collected in 2018 and 2023, respectively. They contain app metadata (e.g., app ID, app name, app category/genre, app description, developer name, etc.), app creatives (e.g., app icons and app screenshots), and for free apps, APK app executables. The 2018 dataset that was collected as a part of our previous work [22] contains metadata of over one million apps that were collected between January and March 2018. The 2023 dataset contains metadata of 1,280,142 apps and was collected between January and November 2023. The data was collected using a Python-based crawler that started from an initial seed of apps, sourced by various top-lists in the Google Play Store progressively discovering new ones. To avoid any disturbances to the app market operation, we conducted our crawl conservatively at a low rate.

i) Top-k apps in 2018 and 2023 datasets - Our key objective in this work is to identify apps that went through notable re-transformations between the two snapshots. To get better insights and to avoid low-quality apps affecting our results as noise, we focus only on the top-k apps in the 2018 dataset and their relevant counterpart in 2023. To identify the top-k apps in each dataset, we use the same method proposed by [18] in sorting the datasets. That is, we sort the set of crawled apps in a dataset (either the 2018 dataset or the 2023 dataset), first based on the number of downloads, second based on rating count, and third based on the average stars and select the first k apps. Next, we constructed a 'validation-set' and a 'test-set' to fine-tune hyper-parameters of the app similarity search algorithm and to evaluate the performance. We discuss our search algorithm later in the Section 3.4.

ii) Validation-sets - When we are searching for a counterpart app between the two datasets, there may be or may not be a matching app. For example, an app may have been discontinued. To properly capture this idea, there must be ground-truth sets on *matches* and *no matches* to fine-tune our algorithm; therefore, we create two validation sets.

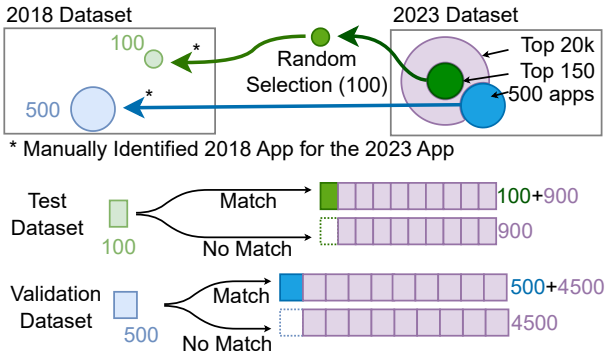


Figure 3.2: Creation of validation and test sets from 2018 and 2023 datasets.

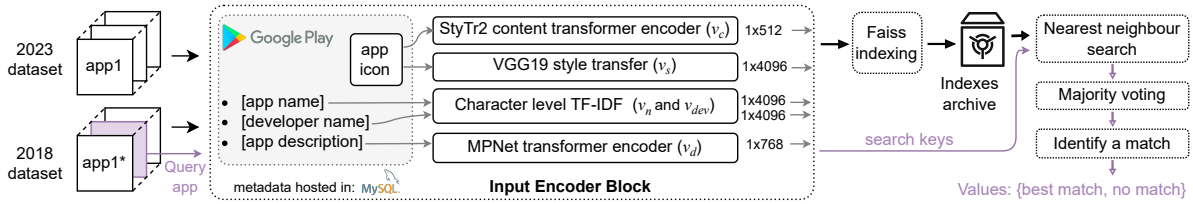


Figure 3.3: Overall methodology for query, key, and value based best match (if it exists - as emphasised in the purple colour path) retrieval for counterpart app identification to an app ID query from 2018 dataset to be matched with indexed 2023 dataset. We utilise Faisis library for creating the indexes archive.

First, we create a validation set for the *match* scenario. As the first step, we select 500 apps from 1,280,142 apps, which has a counterpart in 2023 dataset (teal colour circle in Figure 3.2). We exclude top-150 apps from the 2023 dataset when selecting these 500 apps for test-set creation. We search the counterpart 2018 app, first using the app ID and then manually verifying each one of them (blue colour arrow). The verification is done by cross-referencing the apps' metadata between the two datasets. The figure in Appendix B.3 shows how the top 500 apps are distributed across the 2023 dataset of 1.3M apps. As the next step, we add 4,500 more apps to the selected 500 apps from 2023 dataset, randomly selected from the top-20,000 apps of the 2023 dataset but excluding the first top-150 apps and 199 apps that are selected for the validation set (199 apps are in between top-150 and top-20,000). Now we have 500 apps from 2018 (light-blue colour circle) and 5000 apps from 2023 dataset with 500 guaranteed matching apps between them. Our expectation is to identify all of them as matches during the validation step of our similarity search. To create the *no-match* validation-set, we simply remove the 500 apps from 2023 side that were mapped to the 2018 dataset.

iii) Test-set - To test the performance of our app similarity search algorithm with *matching* and *no-matching* apps, we create two controlled test-sets, quite similar to the method discussed before. We first randomly select 100 apps from top-150 apps in 2023 dataset (green colour circle in Figure 3.2). Then we select their counterpart 2018 app using the app ID followed up by a manual verification (light-green colour arrow). In total, we found that 117 app IDs from

the top-150 apps in the 2023 dataset had exact matches in the 2018 dataset. Next, we add 900 more apps to the 2023 side by randomly selecting from the top-20,000 apps but excluding 199 apps of validation set in the 2023 dataset (purple colour ring in Figure 3.2). This 1000 test-set is used for identifying the accuracy of *matches* of our algorithm during the testing time. To create the *no-match* test-set, we simply remove the 100 apps from 2023 side that were mapped to the 2018 dataset.

3.4 Similarity Search Algorithm

Our objective is to identify a 2023 counterpart app for a given 2018 app (a 'match') or else to return a 'no-match' if such an app does not exist. We follow a multimodal search mechanism to obtain the mappings between the two datasets, and we specifically do not rely on app ID matching in this process. The reason is that metadata-based matching allows us to identify cases of metamorphosis better and to observe whether the developers continued with the same app ID or introduced a new identity.

Our methodology consists of four main steps. ***i)*** *obtain neural embeddings for app icons and app descriptions, and TF-IDF vector representations for app names and developer names,* ***ii)*** *for a given app, retrieve a set of nearest neighbour apps considering each modality,* ***iii)*** *perform modality-wise majority voting to obtain the top-candidate apps for a given app.* ***iv)*** *decide if we could select an app as a 'match' from the candidate app list (further discussed in Section 3.4.4) or else decide it as a 'no-match'.* We show the end-to-end pipeline of the proposed method in Figure 3.3. In the figure, we use the ***query*** app's embedding representations to retrieve results as ***search keys*** and the retrieved results (more specifically, retrieved app IDs) as ***values***. The following subsections explain these steps in detail.

3.4.1 Representation of Different Modalities

As inputs for the search methodology, We use metadata extracted from the Play Store and more specifically, app icon, app description, app name and developer name for both datasets 2018 and 2023. We selected these based on prior work and our investigations on the available metadata that are in spotlight for target app users. (Ablation results in selecting the modalities are depicted in Table B.1 of the Appendix B.4.) For example, developers use the same name to maintain brand consistency for their apps, compared to the developer email, which is often not publicly available and developers might use different email addresses for different apps to avoid confusion.

3.4.1.1 App Icon Embeddings

The app icon serves as the initial point of visual engagement for users, often functioning as a distinctive trademark representing the developer's brand identity. Therefore, a matching pair of apps is likely to have a similar visual 'styling' and similar 'content' information, emphasising the need to encode both such information. StyTr^2 introduced by [205] is a popular transformer-based framework with style and content encoders, both producing output embeddings of the shape (1×512). Additionally, we experiment with vision transformer content encoder embeddings [206] as well as style and content embeddings from VGG19 architecture as proposed in [22]. We identify which style and content embedding combination produces the best results for both 'matches' and 'no-matches' by evaluating possible combinations against our validation dataset. As presented in Table 3.1 we selected the combination of StyTr^2 content embeddings and VGG19 style embeddings to represent app icons.

3.4.1.2 App Description Embeddings

Despite large language generative models being popular, text embedding transformer models, due to their unique encoding capabilities, allow us to obtain rich vector representations for text sequences [207]. Therefore, we use the MPNet [208] model to encode the long app descriptions into the vectors, v_d of shape (1×768). MPNet is a language model similar to BERT [101] and allows to obtain vector representations of text. It considers masked language modelling such as BERT and permuted language modelling such as XLNet [209] in a unified view and thus inherits the advantages of both methods.

3.4.1.3 App Name and Developer Name Embeddings

Since the app and developer names have limited meaningfulness as natural language, as well as they are very short texts, we employ TF-IDF vectorisation to encode app names and developer names instead of using MPNet embeddings. We represent each app and developer name as a vector of size $(1 \times 4,096)$ where the vocabulary contains 4,096 most frequent $1-4$ $grams$ when 200,000 app descriptions were used to build the vocabulary.

3.4.2 Nearest Neighbours of Each Modality

Identifying potential apps that underwent "metamorphosis" involves taking a query app from the top-10,000 apps in the 2018 dataset and identifying the most similar app (if there is any) among the 1,280,142 odd apps in the 2023 dataset. This requires identifying close apps along each modality presented in Section 3.4.1 using nearest neighbour search. However, the vanilla version of the nearest neighbour search is highly inefficient here because of the larger queried set, the number of modalities, and the sizes of the embeddings.

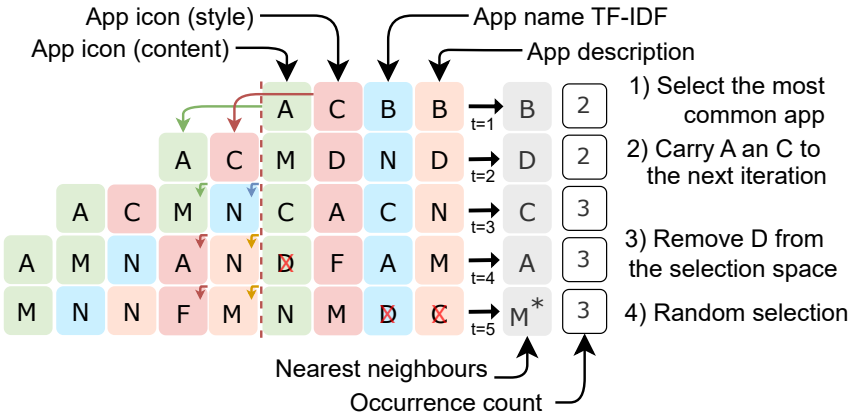


Figure 3.4: Majority voting Algorithm

Instead, we use the indexing options provided in the Faiss library [210] to create an Inverted File Index of the queried embeddings to narrow the search scope. First, we create sharded indexes for small batches of embeddings, and we build the final index on disk by merging sharded indexes into one larger index. We created five such indexes for each modality (represented by an embedding type; app icon - content, app icon - style, app name, app description and developer name), and they are queried for nearest neighbours using the query app's embeddings.

For a single query, we obtain five different nearest neighbour sets corresponding to the five modalities we consider in the descending order of the similarity with the query app's embedding vectors.

3.4.3 Majority Voting

Starting of the app similarity search algorithm is a majority voting scheme based on four modalities (we have excluded the 'developer name' to be discussed later). Figure 3.4 emphasises four main steps based on the previously retrieved nearest neighbour lists for four modalities in the descending (vertical) order. In each iteration, we select the app that has the most number of occurrences in each row-wise-position as the most common result. In the figure, each of these selections is visualised in the grey colour box.

Table 3.1: Harmonic mean on the validation set for different embedding combinations and voting thresholds. (M. = Match)

	$\alpha = 1$		$\alpha = 2$		$\alpha = \mathbf{3}$		$\alpha = 4$		$\alpha = 5$		Harmonic mean				
		No		No		No		No		No	α				
	M.	M.	M.	M.	M.	M.	M.	M.	M.	M.	1	2	3	4	5
StyTr^2_c+VGG19$_s$	88.8	0.0	88.4	36.8	86.0	88.2	63.0	97.0	41.8	99.2	0.0	52.0	**87.1**	76.4	58.8
VGG19$_c$+StyTr^2_s	89.2	0.0	88.8	37.6	85.4	88.2	62.6	97.2	45.2	99.0	0.0	52.8	86.8	76.2	62.1
VGG19$_c$+VGG19$_s$	86.2	0.0	85.8	28.8	83.6	86.6	64.4	96.2	49.4	98.2	0.0	43.1	85.1	77.2	65.7
StyTr^2_s+ViT	90.0	0.0	89.4	35.4	86.2	86.0	65.8	96.4	44.4	98.8	0.0	50.7	86.1	78.2	61.2
ViT$_c$ +VGG19$_s$	88.8	0.0	88.4	32.2	86.2	85.6	70.2	95.6	49.8	98.2	0.0	47.2	85.9	81.0	66.1
StyTr^2_c + StyTr^2_s	85.0	0.0	84.2	35.4	80.6	86.0	54.0	96.4	40.2	98.8	0.0	49.8	83.2	69.2	57.1

As shown in Figure 3.4, in step 1, app B is selected as the most common occurrence since it

appears as the top choice in two separate modalities (occurrence count = 2). Then the remaining apps (A and C) from that position are carried forward to the next iteration. This ensures all potentially similar apps are considered in subsequent row-wise-positions to be iteratively repeated. As shown in step 3, we exclude app D from the selection space since it has already been selected in the second position. When multiple apps appear with the same frequency, we make a random selection. Step 4 indicates this by randomly selecting app M, while both M and N have equal number of occurrences. *The final output is the list of nearest neighbours (vertical list in grey colour) and their corresponding occurrence count for a given query app*.

Note that we handle developer names in a slightly different way as step number 5 that is not shown in Figure 3.4. Since some developers publish more than one app, if we use the developer name as a modality in the neighbour search, the result with the highest occurrence count may not be the desired matched result. For example, if we have the same developer for app X and app Y, then there is no straightforward way to mention X and Y both in the same position in the app developer neighbour list. Therefore, for each row-wise-position, we check the app with the most occurrence count (e.g., app B in step 1) is developed by the same developer of that row-wise-position's app-developer modality result. If so, we consider it as a modality match and add one more to the occurrence count (e.g., app Z in app-developer modality position is the same developer as app B, then we add 1 to the occurrence count in step 1). We only change the occurrence count list (if applicable) in this step.

3.4.4 Match or No-match?

Upon retrieving the list of nearest neighbour apps for a given query app, we identify a *matching* app as the first neighbour app in the list's descending order that agrees with the following two criteria.

- We define a hyper-parameter α such that the number of modalities that contributed for an app to be selected as a nearest neighbour (i.e., the occurrence count) should be equal or greater than α. We call this hyper-parameter as 'occurrence count threshold'.
- We calculate the change of star rating count, representing the number of users who have rated the app, between the neighbour app and the query app. It should be positive. Our intuition here is that a matching app should have grown in rating counts over the past five years.

However, if all the apps in the nearest neighbour search list show a negative star rating count change with respect to the query app, then we only consider the first criteria. But, if no app satisfies the first criteria, we denote it as query app *not having a match*. The threshold value α is selected based on an ablation study we conducted, and we discuss this in Section 3.5. It is also worth noting here that having a *no-match* result does not mean that we exclude that particular app from metamorphosis analysis. We further utilise them to identify and analyse

interesting transitions of apps in Section 3.6.

3.5 Performance Analysis of App Similarity Search

In this section, we evaluate the similarity search algorithm based on the validation and test sets described in Section 3.3. To summarise, when we are using the validation-set for identifying *'matches'*, then we have 500 query apps from 2018 to be matched to a counterpart app among 5000 apps in 2023 dataset. When we are identifying *'no-matches'*, we query the same 500 apps from 2018 to be *'not-selected'* among the 4500 in 2023 dataset because the matching 500 apps of 2023 are removed in this setting. Test-sets are used to observe the performance after the hyper-parameters are selected and they work in a similar setting too; 100 apps in 2018 to match among 1000 total in 2023 for 'matches' and the same 100 apps among 900 in 2023 for 'no-matches'. Next we will discuss the hyper-parameters we validate our model with. We make two hyperparameter choices based on the performance on the validation set.

- We explore six different combinations of style and content embeddings.
- We explore five different occurrence count thresholds (α values) from 1 to 5.

We evaluate a occurrence count threshold, α for each embedding combination and report the harmonic mean of the two scenarios. We show the results in Table 3.1. As can be seen, the combination of StyTr^2 content embeddings and VGG19 style embeddings produce the best performance at the voting threshold $\alpha = 3$. We used this hyperparameter combination to obtain the main results of the chapter.

We also observe that the accuracy of the no-match scenario increases when we increase the thresholds. This is expected because as we increase the thresholds, the number of modalities an app should be included to select as a match increases. On the other hand, the accuracy decreases with the increased thresholds when there is a match. This happens because, as the threshold increases, an app should be included in more modalities in order to be selected as a match, therefore, the conditions become more restrictive. The following subsections describe the performance metrics we use and the performance of the test set.

3.5.1 Performance Metrics

We measure the performance of our method on test set and compare with an existing baseline using accuracy, precision@1 and recall@1 matrices.

3.5.1.1 Accuracy

We take the harmonic mean of the accuracies when there is a *match* and when there is *no-match*. We use harmonic mean instead of the arithmetic mean because we need accuracy in

both scenarios to contribute equally.

3.5.1.2 Precision and Recall

We define Precision and Recall based on the following definitions. For a match scenario, there can be *i) a correct match (true positive), ii) an incorrect match (false positive)*, and *iii) no result when there is a correct match available (false negative)*. Note that there are no true negative occurrences since we are considering matches and no-matches separately. We define the Precision and Recall for a match scenario as in Equation 3.1 and Equation 3.2.

$$Precision = \frac{|True\,Positives|}{|True\,Positives| + |False\,Positives|} \tag{3.1}$$

$$Recall = \frac{|True\,Positives|}{|True\,Positives| + |False\,Negatives|} \tag{3.2}$$

For non-match scenarios, the algorithm can output either no match (True Positive) or an incorrect match (False Negative). Therefore we define only Recall for the non-match as defined in Equation 3.2. Note that Recall and Accuracy are similar for the no-match scenario.

When we report our results as precision@1 and recall@1, we only consider the first matching result for each method when calculating scores.

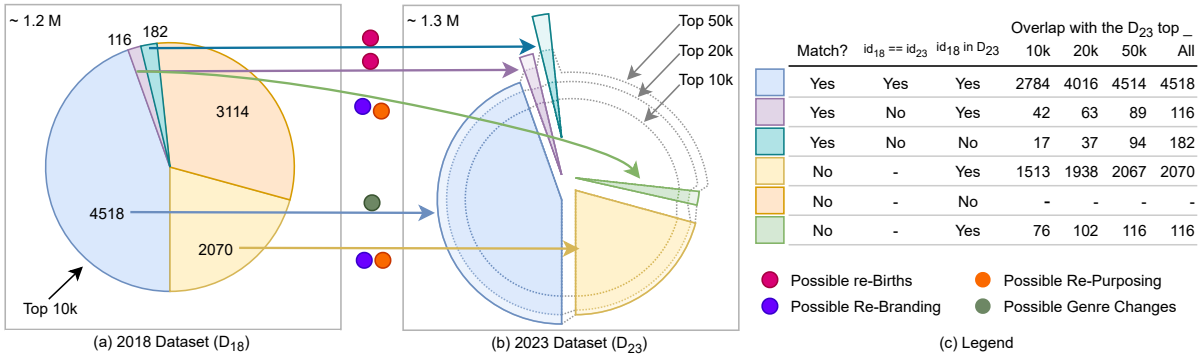
3.5.2 Performance Analysis

We compare our method with the method proposed by Karunanayaka et al. [22]. In [22], the authors proposed a nearest neighbour search for the weighted sum of each embedding, including the app icon (style and content) and app description. We adapt their method using $\alpha, \beta, \theta, \gamma\, and\, \delta$ [22] as the weights for image content embeddings, image style embeddings, developer name embedding, app description embedding and app name embeddings, respectively. In addition to those weights, we use an additional threshold parameter to decide the apps that do not have any matching apps. Then, we choose the optimal values for those weights and the threshold using the Bayesian optimisation [211] on the validation dataset considering the integer values between 1 and 10. We report the results for the best-performed values of $\alpha, \beta, \theta, \gamma, \delta$ and the threshold and results for our method in Table 3.2.

Table 3.2: Similarity search performance on the test set

	Match			No match
	A@1	P@1	R@1	R@1
Majority voting (ours)	**0.930**	0.989	**0.939**	0.900
$\alpha, \beta, \theta, \gamma, \delta$ [22]	0.870	**1.000**	0.870	**0.980**

As can be seen from the results, our method provides better results when the app is included



Match?	id_{18} == id_{23}	id_{18} in D_{23}	Overlap with the D_{23} top _ 10k	20k	50k	All
Yes	Yes	Yes	2784	4016	4514	4518
Yes	No	Yes	42	63	89	116
Yes	No	No	17	37	94	182
No	-	Yes	1513	1938	2067	2070
No	-	No	-	-	-	-
No	-	Yes	76	102	116	116

(a) 2018 Dataset (D_{18}) (b) 2023 Dataset (D_{23}) (c) Legend

Figure 3.5: Identifiable mappings and regions of interests that are obtainable from our similarity matching algorithm. Area under each pie segment is indicative of the proportion of apps belonging to each category as described in the legend. Note: top 10k, 20k and 50k dashed lines are not to the scale, but all pie charts and the proportions they represent between top10k, 20k and 50k are true-to-the scale for better visualisation.

within the search space (i.e., with *matches*). Conversely, the method proposed by [22] demonstrates higher accuracy for *no-matches*. Nonetheless, As shown in Appendix B.2, [22] becomes infeasible to use when the queried dataset and the number of search keys are larger, given that it uses a brute force approach, comparing the search key with embeddings of all apps in the queried data multiple times for different types of embeddings.

3.5.3 Success Score

We introduce *'success score' (SS)* as a parameter that we can use to evaluate the performance of each app progressing from 2018 to 2023. In high-level, we expect a positive SS to indicate an app that has been performing well such that there is no risk of that app going obsolete and a negative SS to indicate an app not performing well.

The success of an Android app could intuitively be associated with the number of downloads and the number of ratings it received from the app users, both ultimately impacting the direct and in-direct revenue incoming to the developers. As Google Play Store only provides end-users with the cumulative download counts and the rating counts as app metadata, it does not reflect current installations (active users) or indicate whether an app is gaining popularity or losing user interest. Therefore, it is relevant to compare the increase or decrease of these numbers across a period of time, as in our study, across five-years. However, an increase in these counts does not necessarily mean an app is performing better due to the natural growth of the Android user base over a period of time: 2.3 billion worldwide devices in 2018 [212] to approximately 3.6 billion in 2023 [213, 214, 215]. Therefore to distinguish an over-performing app and an under-performing app, we need to adjust this natural growth rate which is likely to have been incorporated to any app in the app market.

To address this limitation, we propose a metric, referred to as the success score, which

offsets the average Android user base growth against the average growth in download and user rating counts for a given app. We employ the widely used [216] Compound Annual Growth Rate (CAGR) to analyse the growth or decline of the three key metrics: the number of app downloads, the number of user ratings, and the number of Android device users. Since our dataset spans five-year intervals, CAGR smooths out short-term volatility for a more consistent measure of growth and is ideal for capturing long-term changes by presenting an average annualised growth rate unlike the annual growth rate (AGR) which focuses on year-to-year variations [217, 218].

Considering this factor, we define the **Success Score (SS) for an app based on the CAGR of downloads and ratings offset by the natural growth in the Android ecosystem over a time span of t years**. As specified in Equation 3.3, when we calculate $CAGR_{\#downloads}$, the term $n_{initial}$ is the download count of 2018 app, n_{final} is the download count of counterpart 2023 app and $t = 5$ years. $CAGR_{\#ratings}$ is similarly calculated based on the star rating count and $CAGR_{\#Andro}$ is calculated based on the increase of android devices over the five years, 2.3B to 3.6B.

$$CAGR = (\frac{n_{final}}{n_{initial}})^{\frac{1}{t}-1} \tag{3.3}$$

$$SS = \frac{1}{2}CAGR_{\#downloads} + \frac{1}{2}CAGR_{\#ratings} - CAGR_{\#Andro} \tag{3.4}$$

According to the Equation 3.4, SS value of x indicates that the average number of downloads and user ratings have increased $x\%$ better than the average growth of Android eco-system growth per year. Moving forward, we discuss SS as a percentage (e.g., SS=0.145 is interpreted as 14.5%) for convenience.

3.6 Characterisation of App Metamorphosis

In this section, we provide a characterisation of different types of metamorphosis categories listed in Figure 3.1. We explain how to identify each category using our search methodology, analysis and insights on the causes and effects of metamorphosis, followed by interesting examples.

First, we use our search algorithm on the top 10,000 apps from the 2018 dataset (Figure 3.5a) to identify their corresponding apps among the 1.3 million apps in the 2023 dataset (Figure 3.5b). The legend in Figure 3.5c indicates how we identify the significant metamorphism regions of interest in D18 to D23 mappings. First, we check if our algorithm gives a *match* or a *no-match*. If it is a *match*, we observe if the app ID in 2018 is identical to the app result we get in 2023. If they are not identical (i.e., 2018 app ID $\neq$ 2023 app ID), then we check whether the 2018 original app ID exists in the 2023 dataset; if it exists, we analyse them further.

Lastly, for any of the 2018 top 10k apps that our algorithm gives *no-match*, we check again if those 2018 app IDs exist in the 2023 dataset and analyse further.

3.6.1 Summary of Mappings

As shown in Figure 3.5, a total of 4,518 apps get a *match* in D23 having the same app ID between the two datasets (blue arrow). Nonetheless, only 61.6% of them are still among the Top 10k of D23. The rest were mostly in between the top 10k and 50k. We observe an average success score (SS) of 10.01% for these 4,518 apps, and this score is more influenced by download count ($\sim$20% more contribution) rather than user rating. These apps are likely not to have gone through drastic transformations as our algorithm has already returned a *match*, and the app IDs are the same between the two datasets. We use their SS numbers as the baseline to compare with the SS numbers of different metamorphosis categories.

We observe 298 apps that our algorithm *matched*, without the same app ID among the matching pairs (teal and purple arrows). They suggest that the original app continued to 2023 with a different identity, which we identify as a potential *re-birth* that is discussed later in Section 3.6.2. Out of those 298, 116 had the same app ID linked to other apps in the 2023 dataset (green arrow) that our algorithm did not find as similar. Furthermore, there are 2,070 other examples in the 2018 dataset initially returned again as *no-matches*, but the same app ID exists in the 2023 dataset (yellow arrow). Both of these scenarios suggest that the developer has decided to *re-brand* or *re-purpose* those apps, resulting in major changes to the app description, name and app icon; hence why our methods did not match them. We discuss this further in Section 3.6.3 and Section 3.6.4.

We also discuss other interesting adaptations by the app developers, such as changes in the genre, content-rating, revenue-model and developer-transitions. Furthermore, developers might opt to keep multiple versions of the same app or could develop apps targeted at particular user demographics. We discuss them in detail in Section 3.6.5 to Section 3.6.10.

Finally, there are 3,114 apps, for which our method gives *no matches*, and the same app IDs are no longer present in D23 (orange arrow), indicating either the developers have discontinued those apps or they have been removed by Google (e.g. due to spamming [18]). When an app is discontinued, it is likely caused by the applications of such apps no longer being required or being replaced by progressive versions. Majority of these discontinuations were from tech giants such as *Google, Samsung, HTC* or from popular game developers such as *Gameloft*, and *Electronic Arts*.

3.6.2 Re-born Apps

Re-born apps are characterised by their initial discontinuation, followed by a reappearance where either the original developer or a new one reintroduces the same app concept.

Identification: We select *matched* apps from the similarity search where the 2018 app ID is different from the 2023 app ID. We found a total of 298 belonging to this condition (**cf.** Figure 3.5: 116 in purple and 182 in teal). We further filter them by selecting the apps with a release day in D23, which is later than the last update day in D18, and obtain 88 potential apps for re-birth. We manually validated these 88 apps and found that 74 of them (84.1%) are indeed actual re-births, indicating our identification method works.

Analysis - Because of the app ID change, any market presence the 2018 app gained is discontinued and the new app has to start afresh. As a result, the average success score (SS) for an app in this category is -14.81%, and they struggle to be within the top 10k category in D23 as further illustrated in Figure 3.5 with only 20.4% of the apps represented by purple and teal colours are in the top 10k of 2023. We note the following observations while analysing the re-born apps.

- In Figure 3.7(a), we plot the cumulative distribution function (CDF) of SS scores of re-born apps. It further shows that the success of re-born apps is inferior compared to the baseline SS-CDF of nearly 5,000 matched apps between the datasets, which remain in the top 10k (**cf.** Section 3.6.1). Here, we also observe that nearly $\sim 70\%$ of apps in the re-born category have a negative SS compared to $\sim 32\%$ in the baseline CDF.
- Despite the negative average SS in this category, 30.13% of apps such as, *Bowmasters, MONOPOLY, VLC for Android* have managed to recover (SS>0) and even build upon their user base and ratings within or less than five years time span.
- Games are often ($\sim 27\%$) re-introducing the progressive versions as re-births. An already established user base would actively follow newer versions, therefore contributing to pushing the SS towards a positive value. This allows the developers to discontinue the old version after the transition period to reduce the maintenance cost of multiple apps. Changing the original app to a newer version while re-birthing the former using a new app ID is very rare. But we observed one such example as shown in Figure 3.9(b).
- We identify that sometimes developers change their original app into a totally new type of app using the same app ID (i.e., re-purposed - discussed more in Subsection 3.6.4). However, they do not want to discontinue their original app to keep the revenue stream. They get a new app ID and introduce the original app again, which is a re-birth. For example, *AppLock@DoMobile* changed their *AppLock Theme Nightclub* app to a game called *Block puzzle* and re-introduced the *AppLock Theme Nightclub* under a different app ID (**cf.** Figure 3.9(a)).

- 29.5% of manually verified re-births happen with developer name changes. Even if the app name, description and visual data allow us to verify them as the same app being re-introduced, a malicious actor could easily employ the same strategy and introduce counterfeit apps as re-births when one app gets discontinued. This puts even a tech-savvy user in a vulnerable position as Google Play Store does not provide the history of an app developer. We further discuss this in Section 3.7.

Examples - We have identified notable apps while observing this category, including *Uno* [219] and *Flickr* [220] (**cf.** Figure 3.6). In both of those examples, the original app lost the existing user base as the app ID was discontinued. However, different developers re-introduced very similar apps without a major change in the functionality. In 2019, *SmugMug* acquired Flickr from the former owner *Yahoo* and reintroduced it as a new app with significant changes. However, the SS of the app is -38.8%, indicating that it experienced a significant loss of its user base during this transition. Conversely, the Uno app has a positive SS of 0.7%. Despite a decrease in its rating count, as reflected by a $\text{CAGR}_{ratings}$ of -0.3%, games like Uno naturally has a high demand, evidenced by $\text{CAGR}_{downloads}$ of 20.2%.

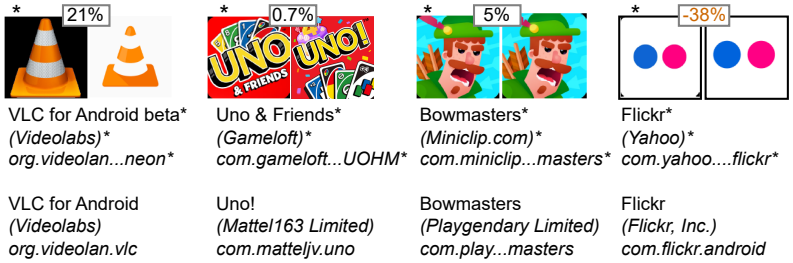


Figure 3.6: Examples for re-birth. Mentioned in italics are the developer name and app ID. (*: indicates 2018 version.) The success score for the transition is numbered inside the text box.

3.6.3 Re-branded Apps

Some apps in Google Play can stagnate after a while because they no longer attract new users, stalling the growth of app-related revenue. In such settings, a developer may opt to change an app's outlook drastically, fine-tune some degree with features or perhaps may opt to transfer/sell the app to a new developer, again resulting in drastic changes to the app according to new ownership. We identify this change as app re-branding.

Identification: We select *non-matched* apps from similarity search where the 2018 app ID still exists in 2023 dataset (**cf.** Figure 3.5: 116 (green) and 2,070 (yellow)). A re-branded app could potentially have changed in the outlook (app icon, visual style and colour schemes, app name) but should have a similar context in the app description. (A drastic change in app description indicates that the app may have been re-purposed rather than re-branded). Therefore, we

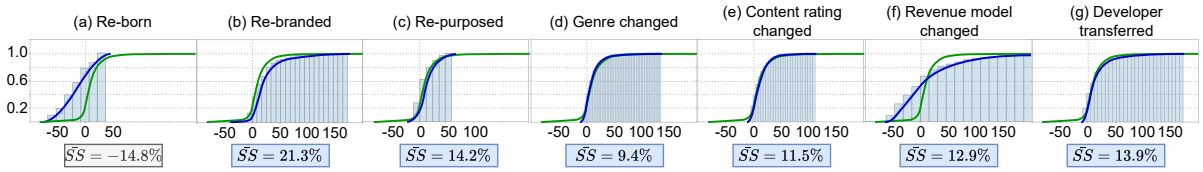


Figure 3.7: CDF plots for the selected metamorphosis categories. X axis represents the success score (SS %). $\bar{SS}$ represents the average SS for all apps undergoing the transition per sub-figure. Legend: Blue: smoothed CDF plot. Green: Baseline CDF for the SS of 4,518 app pairs where our model gives a *match*.

further filter previous results by selecting the apps with app name and app icon content embeddings having a smaller cosine similarity (< 0.7; i.e., these features are likely changed) and app description cosine similarity being higher (>0.4; i.e., still the descriptions remain relatively unchanged) between 2018 and 2023 counterparts.

Analysis: From previous criteria, we retrieved 322 apps that potentially underwent re-branding. Since we know the app ID is identical for all of them across the 2018 and 2023 datasets, there should not be false retrievals. Yet, we manually evaluated all of them for characterisation. 66.8% (215 apps total) of them portrayed an identifiable reason for re-branding; for example, the app name/ app description mentioned the particular changes or the developer was changed along with modifications. The remaining apps had drastic changes in the app name and description, but we were unable to determine a specific root cause. Furthermore, out of the verified 215, 38.6% of the re-branding occurred within the same developer.

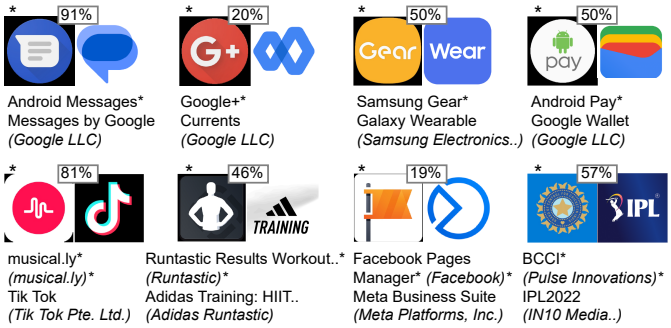


Figure 3.8: Examples for re-branding. (*: indicates 2018 version.) The success score for the transition is numbered inside the text box.

Examples - Our SS calculation for re-brand apps is 21.3% which is notably 11.3% higher than the baseline average SS for ∼5,000 matched apps. This is further emphasised in the Figure 3.7(b) establishing that re-branding has a better chance of success among all other categories of metamorphosis. This positive SS is anticipated as re-branded apps retain their download and rating counts during the transition and re-branding potentially brings new user engagements. Notably, *Mi Drop* to *ShareMe: File sharing* by Xiaomi Inc, *Android Messages* to *Messages by*

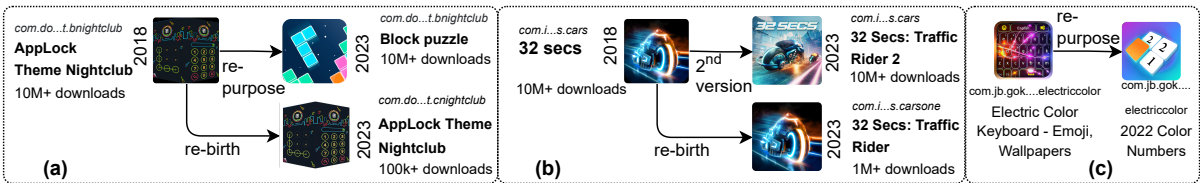


Figure 3.9: (a) a special example where an app re-purposed and a re-birth occurred using a different app ID. (b) an example where an app changed to a different version, however the original app re-born again using a different app ID (c) a generic example of an app re-purposed

Google and *Google Duo* to *Google Meet*, each achieving a SS over 88%. Google LLC claims that the transition of Google Duo to Google Meet is to create a unified platform to provide users with an enhanced experience by adding new Google Meet features to the Duo app. Apps such as *Musical.ly* were re-branded by merging with an existing app like *TikTok* to gain international recognition. Another notable example is Google introducing *Google Currents* as a replacement for *Google+* which has a SS of 10.98%. According to G Suite, this is due to low user engagement and security concerns. Even though Google Current is designed for organisations, the app's main purpose remains the same. This type of overhaul creates new discussion topics among the community to attract more users and also enhance the core functionality the original app was based on. This is quite prominent among tech giants such as *Google, Microsoft* where there is considerable media attention to anything novel coming out of them. As shown in Figure 3.8, other notable examples include *Android Pay* to *Google Wallet*, *Google Fit: Fitness Tracking* to *Google Fit: Activity Tracking* and *Android Messages* to *Messages by Google*. To summarise, re-branding apps tend to show higher success in terms of SS compared to typical apps in the app market, indicating that strategic re-branding can significantly enhance an app's market performance.

3.6.4 Re-purposed Apps

Contrary to what is discussed before, some apps may experience quite a significant loss of interest over time and may no longer provide useful services to the user base. For example, rapid developments in the Android operating system could cause many popular apps such as camera, file management, battery optimisation and other utility apps to become obsolete quickly. The developers may want to relaunch them by overhauling the original functionality; we call them "re-purposed" apps. The basic intuition behind this incentive is that the original app has a significant user base and contains good reviews, and the developers need to build on top of that. There could also be instances where a developer needs to transfer the app to a blooming app category, away from the existing competition but intending to maintain the previous user base. Overall, reasons to re-purpose an app compared to the original app concept could be summarised as follows;

- The developers perform drastic changes in core functionality; therefore, the original app is re-purposed, and the original concept is now discontinued. This can also be considered as an extreme end of app re-branding.
- The developers re-purposed the original app. However, they intend to retain their original concept and reintroduce it as a re-birth (**cf.** Section 3.6.2).

Identification: How we identify re-purposed apps is the same as identifying re-branded apps. However, we only retrieve instances where the app description's cosine similarity is between 0.2 and 0.4. The reason is that a re-purposed app drastically changes functionality, and the app description embeddings should deviate significantly from one another. Similarities of less than 0.2 were observed to be noisy, such as descriptions in two different languages, and as a result, we discarded them from the evaluation. The selection criteria is summarised in Appendix B.5.

Analysis: The average SS for this category is 14.3%, which is lesser than the SS of re-branded apps yet higher than the baseline as shown in Figure 3.7(c). This could be due to the nature of re-purposing, which involves significantly changing the app's functionality or target audience, which can negatively affect its existing user base. In contrast, re-branding maintains its core functionality and changes the brand elements. Overall, we identified 33 re-purposed examples, and we manually verified all of them. We identified 24.2% of them as ideal examples. We couldn't conclusively verify the rest of the apps, which will likely be at the intersection of re-branding and re-purposing.

Examples - We show some examples of re-purposing in Figure 3.9(a) and Figure 3.9(c). The *AppLock Theme Nightclub* app was re-purposed to a *Block Puzzle* game, while *Electric Color Keyboard* was re-purposed in to *2022 Color Numbers* apps. Both of these examples highlight how developers leveraged their existing user base to introduce new app concepts. Some other notable examples include; *Fingerprint Lock Screen - Prank* (10M+ downloads) re-purposed to *GPS Navigation - Route Finder* (10M+ downloads) (app ID: `com.galaxyapps.lock`). An incentive from developers like this is undesirable for app users as they did not download the original app for a navigational purpose.

3.6.5 Genre Changes

Genre changes, also known as category changes, represent the instances where the high-level category (e.g., Educational, Simulation, Casual) is changed in an app. We obtain this information from the app metadata hosted in Google Play Store for each app.

Identification - We discover a total of 1,111 alterations in the app categories among the 4,518 *match* and *2018 app ID = 2023 app ID* instances identified by our algorithm. We present the

most commonly observed category changes in Figure 3.10(a).

Analysis and Examples - The average SS for apps that have undergone a genre change is 9.44% as shown in Figure 3.7 (d). Therefore, despite the app undergoing genre changes, it remains as competitive as an average top 10k app on average. As shown in Figure 3.10(a), the most frequent genre transition involves casual apps shifting to simulations. Even though 209 apps have made this transition, their SS is lower than the average at 7.74%. However, 23 apps that became simulation from being role playing, have achieved an average SS of 22.14%.

The motivation behind altering the app category/genre by developers can stem from several plausible reasons, including i) the aim to better align the app with its features, functionality, or target audience; ii) adjusting to app market changes (e.g., from "Candy Crush Saga" categorized as "Casual" to "Puzzle"); and perhaps iii) move the app by modifying the app category to go into a less competitive app category or to avoid scrutiny (e.g., "CVS Pharmacy" categorised as "Health & Fitness" to "Shopping"). There has been some evidence for this behaviour in the past [204].

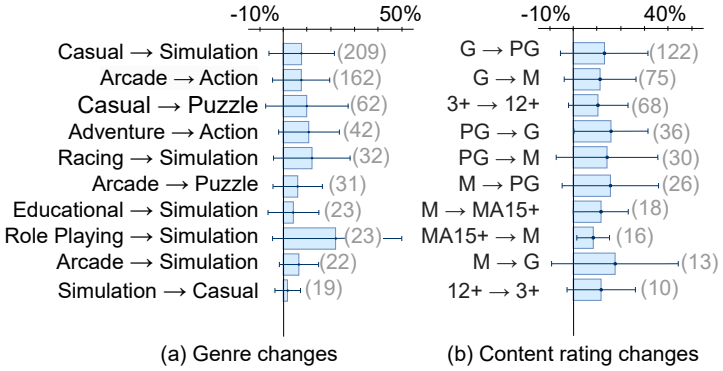


Figure 3.10: Diagram of 10 most common genre (a) and content rating (b) changes with respect to their average success score. Standard deviation of the SS for each transition is also shown using error-bars. The number enclosed in parenthesis represents how many apps underwent the transition.

3.6.6 Content Rating Changes

Content rating, also known as age rating in Android, is self-reported by the developer. This information is also available in the app metadata in the Play Store. In Australia, where we conducted the data collection, games follow a content maturity rating issued by the local regulatory body (i.e., G, PG, M, MA15+, and R18+), and other apps follow the International Age Rating Coalition (IARC) rating (i.e., Ages 3+, 7+, 12+, 16+, and 18+). Over the past five years, we observe a noticeable shift in content ratings of apps.

Identification - We were able to identify 504 apps with different content ratings among the

4,518 *match* and *2018 app ID = 2023 app ID* instances. We present the most commonly observed changes in Figure 3.10(b).

Analysis and Examples - Overall SS for the 504 apps that have changed the content rating is 11.54% as shown in Figure 3.7 (e). Within this, we identified 11 apps that has been re-born with changes in content ranting, for example *Uno* shifted its content rating from G to PG after being taken over by a new developer. Further, 17 apps became games and 12 games transitioned into generic apps. However, upon manual analysis, we found that those 17 apps that became games were games in the first place, and the developer content rating was not correctly reflecting that. As an example, the game 'Tanktastic 3D Tanks' in 2018 was rated for 3+ and in 2023 (giving a false illusion that this could be a generic app instead of a game), the rating changed to PG (Parental Guidance) and a better match for the overall theme of tank battles.

We observe a similar setting among game to app transitions as well. Furthermore, as shown in Figure 3.10(b), the most common change happening among game apps is 122 of them increasing their rating from G (General) to PG. A popular game of 'Mafia City' changed from PG to M due to violence, strong language and blood-related themes in the game.

We attribute the changes discussed above as better alignment with Google Play Store rating guidelines and more developers being aware of any previous mistake they did. Further restricting a content rating category than the original one is not necessarily adverse. However, whether the existing users were duly notified is questionable. On the other hand, loosening the content rating category could attract more popularity among a wide range of users if they comply with content-rating regulations. Among the 26 games that changed the rating from M to PG, we identified 14 of them directly contained Gun violence related app icons, which is questionable. Since identifying content compliance malpractices is not our primary concern, we open this topic for future research.

3.6.7 Changes in Revenue Model

In this context, we consider only the direct revenue model where an app is either "free" or requires payment for installation (hereafter referred to as "paid apps").

Identification - This information is extractable directly via app metadata, and we can utilise our similarity search algorithm to obtain matching pairs and identify the instances of paid apps becoming free. In 2018, there were 38 paid apps, however 9 of them discontinued by 2023.

Examples - Among the results we obtained, we did not observe any examples within the top 10,000 apps of the 2018 dataset that changed from a paid app to a free app. It is understandable that popular paid apps may not consider moving to a free model, as they would not want to lose their existing users who bring direct revenue. Nonetheless, upon further investigation using the entire 2018 dataset, we could identify 649 paid apps in 2018 that have become free apps with an

average SS of $\sim 13\%$ (cf. Figure 3.7-f). A noteworthy example is *TextGrabber – image to text: OCR* where originally it was a paid app ($6.99) but was later changed to a free app, possibly due to many similar tools being available elsewhere for free and the developer deciding freemium might be a better business model than simply offering a paid app. Moreover, *Farm Simulator 16* transitioned to a free app and its download count jumped from less than 1M in 2018 to 65M, securing a top spot in 2023.

3.6.8 Transferred Apps

There is an option in Google Play Store for an app to be transferred from one developer to another developer [221]. Such transfers may occur when developers adopt new business models centered around rapidly growing mobile app concepts (e.g., spin-offs) or when their businesses are acquired by different companies.

Identification - To identify these types of apps, we employ our search method to find *matches* where 2018 app ID = 2023 app ID. From these selections, we extract the ones with a cosine similarity of the developer's name, email, and website below a threshold of 0.5. In this way, we identified a total of 104 potential app transfers with an average SS of $\sim 14\%$ (cf. Figure 3.7-g). *Note: This is not to be confused with 'Re-branding' where our algorithm does not give a 'match' due to the considerable change in app's appearance. Here, it is simply handed over without major changes.*

Examples - *Subway Surfers*, the first game to cross 1 billion downloads on the Google Play Store, was initially co-developed by both *Kiloo* and *SYBO Games*. However, *Kiloo* has withdrawn from the development and the game is now solely owned by *SYBO Games* [222]. Another example is shown in Figure 3.1 where *Flip Master* game is transferred from popular developer *Miniclip* to *MotionVolt Games*. Overall, app transfers between developers can bring fresh perspectives, resources, and opportunities for growth, ensuring the continued development, support, and success of such mobile apps.

3.6.9 Changes in Target Demography

Some developers may opt to publish different variations of the same base app for different target audiences. In previous subsections, we retrieved only the top result of the 5 nearest neighbours by our similarity search algorithm. Observing the results for the remaining nearest neighbours with the condition developer name 2018 $\approx$ developer name 2023, we observe apps belonging to this category.

Age - There are 84 apps that are targeting users of different age-demographics with the same

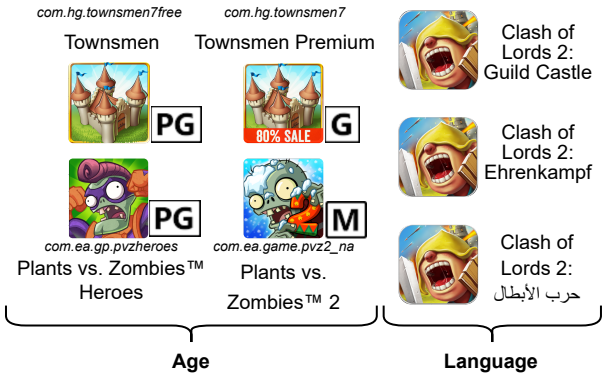


Figure 3.11: Some examples of apps where the target demography changed based on age and language.

Figure 3.12: Progressive version examples for two apps, *Cut the Rope* and *Bubble Witch Saga*.

concept. Examples include apps such as *Plants vs Zombies™ 2* (Mature) and *Plants vs. Zombies™ Heroes* (Parental Guidance), *Vector 2* (General) and *Vector 2 Premium* (Parental Guidance), *Silly Sausage in Meat Land* (Mature) and *Silly Sausage: Doggy Dessert* (General).

Language and geographical region - We noticed 21 apps with multiple languages or target localities. For example, *Clash of Lords 2* has multiple versions for different languages as shown in Figure 3.11 and *TikTok* has two different versions; *com.ss.android.ugc.trill* and *com.zhiliaoapp.musically*. The former version primarily targets East Asian and Southeast Asian countries, while the latter is intended for other countries.

Multiple revenue models - Exapanding the analysis from previous subsection about changes to the original revenue model, we observed 88 apps that exhibit multiple price variations, with some being offered as free apps and others requiring a payment. A notable example is Need for Speed^{TM} Most Wanted with a paid app; *com.ea.games.nfs13_na* and a free app; *com.ea.games.nfs13_row* (only available for some mobile devices).

3.6.10 Progressive Versions of Apps

The decision to release newer versions of an app gives opportunity for the developers to enhance features, address bugs, and meet evolving security and technological requirements based on user feedback and suggestions. We discussed in Sub. Section 3.6.2 that most of the games abandon

previous versions after some time when the user base has shifted to the newly released version. However, some developers may opt to maintain retain the previous versions such that they can target different niche markets, address specific platforms or hardware compatibility issues, or experiment with novel concepts. Later, based on user feedback and market response, the developer has the flexibility to decide whether to continue maintaining both versions or discontinue one of them. We observe 380 examples of 2018 top 10,000 dataset now has multiple versions by observing the nearest neighbours of our similarity search algorithm with the condition of developer name 2018 $\approx$ developer name 2023. We highlight some examples in Figure 3.12.

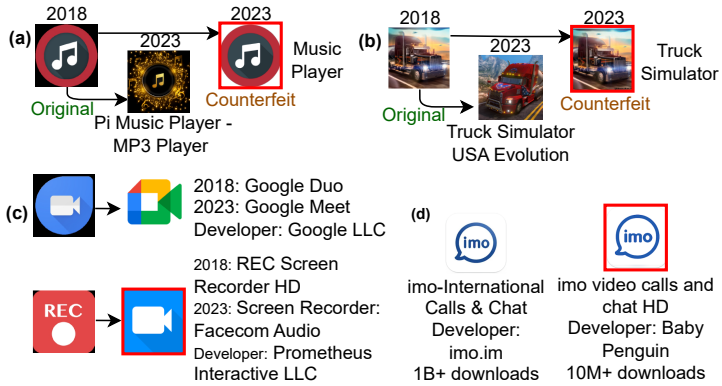


Figure 3.13: Security Risks of Mobile App Metamorphosis. Outlined in red are observed to be counterfeits/ practicing plagiarism.

3.7 Privacy and Security Implications of Metamorphosis

Even if the general belief is that the top apps in Google Play Store are trustworthy and are better moderated, our results showed that an app's visual appearance, textual description, developer name, or all of them could change under metamorphosis. Some metamorphosis scenarios involve cases where the app ID remains unchanged, and updates are pushed to users who have already installed the app. The behaviour, branding and functionality changes can come as a total surprise to the user. Also, at Google Play Store, potential new users will not see any transformations an app has undergone and are compelled to *"download an app as you see it"*, even if we remember *"what it used to be"*. We next discuss and present further results to emphasise why app metamorphosis is an important phenomenon that requires attention in app markets.

3.7.1 App Counterfeits and Plagiarism

In two instances out of 88 potential re-birth instances that were flagged by our methodology, we observed that the original app was discontinued and re-born with a different app ID. Further analysis established that the original apps did not discontinue, instead they went through some major changes.

For example, Pi Music Player was renamed to Pi Music Player- MP3 Player with drastic changes in the app icon and other metadata as portrayed in Figure 3.13(a). Noticing this change, a counterfeit app emerged in the name of Music Player with the same app icon that used to be with the authentic app and reached nearly 100k+ downloads. Similarly, a counterfeit app called Truck Simulator appeared when the original app Truck Simulator USA was changed to Truck Simulator USA Evolution, as shown in Figure 3.13(b). This counterfeit app, however, is now removed from Google Play Store but managed to reach $\sim 200k$ downloads.

In one instance of app re-branding, when Google Duo changed to Google Meet, another app developer picked up an almost similar app icon to the discontinued Google Duo app, aiming to trick users into downloading the app based on icon familiarity. This is shown in Figure 3.13(c). Another instance, Figure 3.13(d), we observed is a developer named Baby Penguin maintaining an app visually similar to the popular imo-International Calls & Chat app, potentially victimising more than 10 million users, until it was discontinued/banned from Play Store.

It is important to note that we were not actively looking for counterfeits as they are outside our main research focus. Rather, we found them to be a byproduct of our metamorphosis analysis. Nonetheless, results like these show the security and privacy risks even to the most tech-savvy users in identifying the authenticity of an app among the top apps in Google Play Store. The examples we observed did not contain any malware indications when evaluating their installation files with VirusTotal service, rather observed as an effort to tap on to existing popular apps in top 10,000 category. While previous work has developed methods for app counterfeit detection when both the target and the counterfeit apps are in the market and highlighted the associated associated risks, such as counterfeit apps being channels for malware distribution [22], our metamorphosis analysis provides insights into how other scenarios lead to counterfeits.

We encourage app market operators to deploy a similar pipeline as we proposed and actively investigate new additions to the app market. As emphasised in Appendix B.2, search efficiency improvement of our method further enables it to be deployed in the real-world scale where an app is introduced every 1.13 minutes to the PlayStore [223]. We suggest the app markets to observe the resultant matching app(s) if existing and bin the results to appropriate metamorphosis categories we identified. Ideally, the operators can perform this for multiple past-indexed years to obtain an even finer-granular picture. The market operators have a further advantage to identify if a matching app is counterfeit or not as they have the relevant history for a particular change (e.g. the developer changed the name, the developer re-branded the app, etc.) and further investigate any other potential red flags. Though it is out of our scope, malware detection techniques (e.g. VirusTotal) and static or dynamic code analysis techniques are resource intensive and could not be done for every new app. Instead, any red flags from our method are ideal to be selected for such additional checks. We believe the metamorphosis concept we

propose simplifies finer-granular investigation steps for market operators. On the contrary, the researchers and end-users hardly have information on the history of an app or an app-developer unless it is covered by media such as with music.ly to tiktok example.

3.7.2 Permission Changes

We evaluated the changes in the permissions of the manually verified apps belonging to re-birth (74), re-branding (215) and re-purposing (8) metamorphosis categories and their percentage changes are shown in Figure 3.14. We categorised each permission based on the risk associated with it as previously suggested by [224].

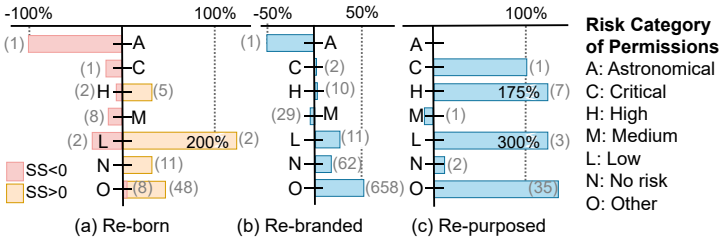


Figure 3.14: Percentage change of permissions according to the risk category when the apps underwent three metamorphosis categories. (x) indicates the number of permissions that changed.

When an app undergoes a re-birth, it is essentially a new app; therefore, a change of permissions is expected. Among the 74 apps, we observe that astronomical and critical permission requests have been reduced. Re-branded apps and re-purposed apps, on the other hand, are essentially the same apps if the user installed them before the transition period as there is no change in app ID and the app will be updated automatically (based on the user settings) and new permission requests will be prompted to the users.

With re-branding, we do not observe significant permissions growth for medium and above risk categories. Comparatively, the re-purposed apps have a significant growth of permission requests for high and critical. It is questionable whether existing users are transparently informed about when apps undergoing major changes that re-purpose their core functionality, as users tend to routinely allow permission prompts without much scrutiny. Therefore, we reiterate the importance of users remaining vigilant about routine permission requests from apps and carefully evaluating their necessity.

From a privacy standpoint, majority of end-users are expressing concerns about data collection and retention practices, but there is an ambiguity surrounding how sensitive information is handled during app metamorphosis. While Google Play places a strong emphasis on developers disclosing their practices, specific instances of app metamorphosis, such as when an app changes its developer (app-transferred), are not explicitly addressed on the Google Play Store. Consequently, most app users are unlikely to notice such changes and remain unaware of how

their information is being passed along. Therefore, it is crucial to carefully observe privacy change notices when they appear while using an app.

Similarly, app market operators should also implement mechanisms to detect changes in app permissions that may occur during legitimate app metamorphosis. When significant changes are identified, such as changes in authorship or a large number of new permissions, users should be notified promptly. These notifications should clearly inform users that the app has undergone substantial modifications and that caution is advised before continuing to use it.

3.7.3 Breach of User Trust

Developers are encouraged to provide routine updates to the apps and as a result, they have the freedom to change app matadata to reflect such changes. In fact, there are apps that can benefit from changing the theme according to the season, such as snowy themes during the winter trimonth (e.g., shopping apps and games providing holiday-themed in-app purchases). However, due to the freedom of metadata change, we observed that the genre category (ref. Section 3.6.5), content rating category (ref. Section 3.6.6) and even the revenue model (ref. Section 3.6.7) have been changed while the app was not undergoing major reformations in functionality or appearance. We have identified several factors that could have influenced these changes, yet this creates a breach in user trust. A game could change the content rating from PG (parental guidance) to M (mature), however, a child who downloaded the game previously could be exposed to sensitive content not intended for them, and often the developers could collect blind consenting in the form of "We are updating the app" report that users rarely read. Similarly, a game that changed from "puzzle" into "simulation" where the original users expected it to be "puzzle" is again a breach of their trust and a waste of their time. An app becoming free after being a paid app should consider having refund policies for their existing users, yet most of such apps being one-time payments at the beginning makes it nearly impossible to refund. In such a context, it is important that app market operators be more vigilant about these metamorphosis categories and promptly note their existing user base about these particular changes, in addition to what the developers note.

3.8 Concluding Remarks

Using our proposed multi-modal app search methodology, we investigated the phenomenon of *"app metamorphosis"* occurring on the Google Play Store between two snapshots taken five years apart. To the best of our knowledge, this is the first study examining this phenomenon. By defining a success score (SS) for each app, we quantitatively characterised the metamorphosis categories. Our observations revealed that although these apps do not follow the traditional app life cycle of incremental updates, the majority of apps in these categories are more successful

compared to an average app within the top 10,000 category in 2018.

We found interesting forms of metamorphosis such as re-birthing (average $SS \approx -15\%$), re-branding (average $SS \approx 21\%$), re-purposing (average $SS \approx 14\%$), for which we provided details on identification and analysis. In addition to these we also found other forms of metamorphosis such as genre changes, content-rating changes, revenue model changes, developer transfers, targeting multiple demographics, and coming in multiple versions. We observed that re-branding is more successful than the other categories.

While being restricted to only two snapshots is a limitation, our focus was on identifying significant cases of 'metamorphosis' over a period of five years. Introducing an intermediate snapshot could shift the focus toward smaller, incremental changes, which deviate from our primary goal. For example, re-branding is often a gradual process involving changes such as app icons, descriptions, or minor UI adjustments. An intermediate snapshot might introduce ambiguity between seasonal re-branding, incremental re-branding, and full re-branding, complicating detection. Similarly, for re-birth scenarios, where a re-born app co-exists with its old version temporarily, an intermediate snapshot could blur the classification, making it unclear whether to treat this as a re-birth or a different version entirely. However, we were still able to conclusively validate the majority of our results. For example, out of all potential rebirth and re-branding results, we manually verified approximately 85% and 78%, respectively, to be conclusive and strong examples.

Additionally, we presented insights into privacy and security risks arising from metamorphosis, such as app counterfeiting, requests for additional permissions, and changes in app behaviour. Our results show that app ecosystems need to pay close attention to the metamorphosis phenomenon to maintain the integrity of their stores. Based on our proposed methodology, market operators can analyse app metadata changes in a similar way how app users perceive information prior to installation, and based on observable metamorphosis categories (if any), they can further analyse latest versions against possible past versions and flag any mismatches. Market operators have much further capacity to view the history of app developers, perform static and dynamic code analysis to further evaluate such flagged instances. The flags created by our method will aid in narrowing down this extensive analysis process to the metamorphosis apps that could be potentially risky and operators will have the capacity to take them down even before user complaints.

Chapter 4

Detecting Content Rating Violations in Android Applications: A Vision-Language Approach

In the previous chapter, our analysis centred on the broader evolutionary dynamics of mobile applications rather than on conventional incremental development patterns. In this chapter,[1] We shift our focus to the problem of detecting content rating malpractices in mobile apps. Specifically, we investigate a vision language framework that leverages multi-modal contrastive learning to implicitly capture both stylistic and semantic cues present in app metadata.

4.1 Introduction

In recent years, our reliance on mobile services has surged, whether for entertainment, communication, shopping, travel, or finance. According to recent reports, 60.42% of the world's population owns a smart device [225]. One implication of this trend is that children of all ages are using smart devices and apps more than ever.

In 2013, a survey revealed that over 75% of children under 8 years old were using mobile phones [71] and in 2019, 69% of teens owned a smartphone [10]. This widespread dependence among young children poses a significant social challenge whether they are being provided with age-appropriate content, usually defined by content ratings. For example, the Google Play Store ratings in the US and Canada are maintained by the Entertainment Software Rating Board (ESRB), with rating categories *Everyone, Everyone 10+, Teen, Mature* and *Adult only*, whereas in Australia, games adhere to local content maturity ratings issued by the Australian

[1]This chapter incorporates material from: **Dishanika Denipitiyage**, Bhanuka Silva, Suranga Seneviratne, Aruna Seneviratne, Sanjay Chawla, A Vision–Language Approach with Cross-Attention for Detecting Content Rating Malpractices in Android Applications, accepted at the IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2025)

Figure 4.1: Example predictions of two apps: Bingo Mania (left) and Cute Cats Jigsaw Puzzle (right). Our model correctly identifies content rating, while GPT-4o under-predicts both, assigning lower classifications.

Classification Board (ACB), including G, PG, M, MA15+, and R18+. Similarly, rating systems are employed in different regions, such as Pan-European Game Information (PEGI) in Europe. These guidelines use six different content descriptors: themes, violence, sex, language, drug use (substances) and nudity and asses the impact depends on the frequency and/or intensity.

To assist users, especially parents, in finding suitable apps, the Google Play Store enforces strict developer policies. As a result, each app page displays critical app information, such as download counts, requested permissions, content rating, developer names, and user reviews, allowing users/ parents to understand the app before downloading. Additionally, all apps undergo automated inspection and vetting procedures before being published. Furthermore, in 2020, Play Store introduced "Teacher Approved" apps, which are published after being rated by teachers and specialists. They take into consideration factors such as design, age appropriateness, and appeal when rating an app [226]. Despite such initiatives, Luo et al. [227] identified that 40% of apps contained inappropriate content among 70 children's apps in 2020. Moreover, children reportedly spent 27% more screen time on online video platforms, 120% more for TikTok in 2023 compared to 2022 [228] despite those apps being rated for ages 13 and up. Furthermore, by analysing 20,000 apps in Google Play Store, Sun et al. [47] claimed that 19.25% of apps have inconsistent content ratings across different protection authorities around the world, thus making them un-generalisable.

One reason why such content rating violations and discrepancies are possible among apps, especially in the Google Play Store, is its loosely regulated nature. The Google Play Store relies on an app developer's completed questionnaire and self-reported information to automatically determine the content rating as disclosed by developers [229]. In a profit-driven app ecosystem with over 3.6 million apps in the Google Play Store [214], where app engagements matter

significantly, especially those from young children, it can not always be expected that all developers will play fair. Furthermore, Google employs different content rating systems based on geographical locations [229], and there are no clear boundaries among the categories, to the extent that an average smartphone user can easily get overwhelmed. On the other hand, the app vetting process by the Apple App Store is manual [73], and as a result, it is most likely to have correct content ratings for apps; however, the downside is that getting apps published in the App Store takes time.

Even advanced general-purpose models struggle to accurately classify app content ratings. For instance, as shown in Figure 4.1, GPT-4o, despite having access to both the app description and game image, misclassifies apps containing simulated gambling or strong horror themes as suitable for general audiences. These examples highlight the complexity of this task: in the first case (left) [230], the app contains simulated gambling, yet this detail is not explicitly mentioned in the text or easily inferred from the app image. In the second case (right) [231], the app includes strong horror themes and online interactivity, but the description presents conflicting cues, stating it is designed for both adults and children, while also encouraging users to share images via the internet and social media. Such ambiguity and lack of explicit content warnings make it difficult even for powerful models like GPT-4o to make correct predictions, emphasising the need for a task-specific model trained to detect the content rating of mobile apps.

As such, there is a stronger need to develop methods to automatically assign correct content ratings for given apps. This requirement is further exacerbated by the fact that regulatory bodies, such as a country's e-safety commission, do not have the necessary means to identify apps that violate the country's content rating guidelines unless end users complain about specific apps. Currently, such studies by regulators are mostly carried out manually or semi-manually. For example, in 2012, the FTC reviewed 200 apps, mostly through manual processes [232].

To this end, in this chapter, we propose a vision-language approach based on self-supervised learning and supervised contrastive learning that allows us to identify content rating violations in app markets. Our approach is based on the intuition that multi-modality is important in this problem (i.e., considering both app descriptions and images such as icons and screenshots, commonly known as app creatives). Within creatives, it is crucial to consider both the content and style of these images. The style information is effective in identifying the target demographic of an app, as apps designed for children tend to have cartoon effects and tactile textures like glitter. More specifically, we make the following contributions:

- We propose a vision-language approach using trained content, style, and text encoders, along with a cross-attention module, to predict mobile app content ratings from descriptions and creatives.
- Using a real-world dataset, we show that our approach achieves 8.12% and 7.99% relative improvements in accuracy over state-of-the-art CLIP and CLIP fine-tuned models.

- Upon evaluating the test dataset, our model identified 71 apps (∼17% of the total verified) with potential content rating malpractices in the Google Play Store and 32 apps subtly attracting an unsuitable audience. This includes nine 'Teacher Approved' apps, which Google Play claims to verify manually.
- We conduct extensive experiments on nearly 16,000 apps to validate the effectiveness of our model. The results show that 45.7% of identified malpractices and 39.1% of identified disguises were removed by the Play Store after nine months of our initial crawl.

4.2 Related Work

Automatic app maturity ratings: The evaluation of mobile apps often involves various perspectives. In particular, identifying mobile app development is consistent with what is stated in the privacy policy concerning online advertising and tracking [85, 233], aiding developers in crafting child-friendly apps concerning both content and privacy aspects [55, 234]. However, fewer studies aimed at mobile app maturity rating. Therefore, there is growing concern regarding inappropriate content and maturity ratings in mobile apps, which are linked to privacy concerns. Early work by Chen et al. [53] proposed Automatic Label of Maturity ratings (ALM), a text-mining-based semi-supervised algorithm that uses app descriptions and user reviews to determine maturity ratings. The authors used the content rating from Apple App Store as the reference standard for a given app. However, this method uses keyword matching while ignoring semantic analysis. Using a similar approach for ground truth establishment, Hu et al. [55] proposed a text feature-based SVM classifier for content rating prediction with an online training element. The previous two methods solely depend on text features despite having access to other modalities. Liu et al. [71] and Chenyu et al. [54] extended these works by incorporating image and APK features to identify children's apps. However, features were limited to extracting text using OCR software, colour distribution of the icon and screenshots, and permissions and APIs. More recently, Sun et al. [47] identified discrepancies in content ratings of the same app in different geographic regions by defining rating system mappings between geographical regions. However, this research focuses on single modalities or multiple modalities but treats them independently.

Vision-Language (VL) models: Early image-based contrastive representations have made advancements, nearly achieving the performance levels seen in supervised baselines across various downstream tasks such as image classification and retrieval [116, 235]. Driven by the success of contrastive learning in intra-modal tasks, there has been a growing interest in developing multi-modal objectives (e.g., Vision-Language), enabling the model to comprehend and exploit cross-modal associations. Pioneering works such as CLIP [50] and ALIGN [51] bridged the gap

between the vision and language modalities by learning language and vision encoders jointly with a symmetric cross-entropy loss which is an adaptation of InfoNCE loss [236] for cross-model pairs. CLIP optimises the cosine similarity between text and image embeddings, while ALIGN employs a similar contrastive learning setting with noisy training data. Zhai et al. [237] tuned the text encoder using image-text pairs while keeping the image encoder frozen. The rich embeddings that these methods learn are later adapted to various application domains such as video-text retrieval [238, 239], image generation [240], and visual assistance [241]. However, [242, 243] point out the challenges in adapting Large Multi-modal Models (LMMs) for different domains when the downstream task deviates from the originally pre-trained task. To the best of our understanding, ours is the first work to leverage the advances in VL-language models to detect content compliance malpractices specific to mobile apps.

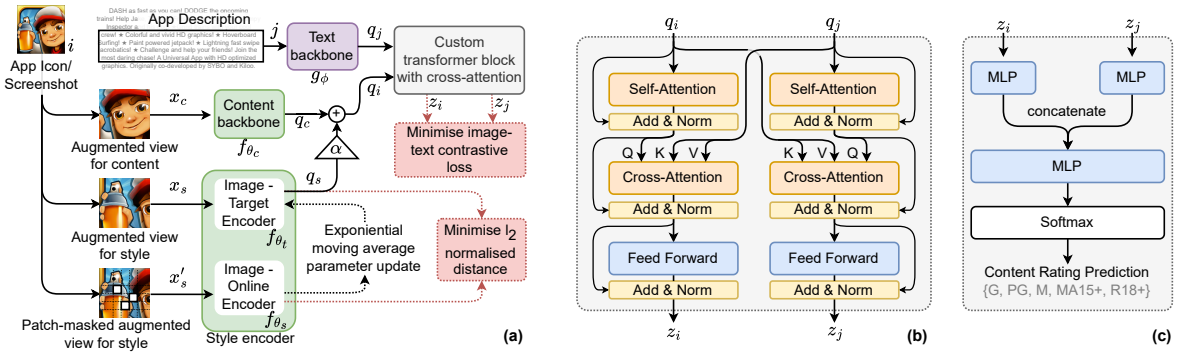


Figure 4.2: (a): Vision-language model architecture during the training stage. (b): Custom transformer block with cross-attention. (c): Pipeline for the downstream task of content rating classification.

4.3 Methodology

We propose a customised vision-language (VL) model that can be trained end-to-end with image and language data for learning joint representations directly via image patches and raw text tokens. Figure 4.2(a) gives an overview of our model. It uses mobile app creatives such as app icons/ screenshots and app descriptions as the paired inputs to generate joint embeddings that are useful for the downstream task of content rating prediction. We discuss our dataset in detail in Section 4.4.1.

At a high level, we adapt two image encoder-based backbones to learn image content and style features separately. These two encoded features are merged together as image features. Then, we employ a text backbone to encode text features in the corresponding image-text pair belonging to an app. A cross-modal module then extracts relationships between image and textual features to produce the final image and textual embedding representations. We use pair-wise Sigmoid contrastive loss to learn the model parameters. Using the trained text and

image embeddings, we then train a simple classifier to predict the content rating of an app, as illustrated in Figure 4.2(c).

4.3.1 Encoding Visual Information

App icons and screenshots are the most prominent static-visual information the prospective app users first observe. In some cases, the style of an app icon alone is enough to distinguish many popular apps. As an example, app icons from Google would likely contain four colours: red, yellow, green and blue (**cf.** Figure 4.3). However, encoding such information is challenging as they do not contain features a generalised encoder such as CLIP was pre-trained on. To address this, we introduce trainable content and style encoder modules that work together to generate the final image embedding vector. Their ablation studies are further discussed in Section 4.5.3.

4.3.1.1 Style Encoder

Style information (e.g., texture, colour schemes, artistic choices) is crucial when predicting content rating as it adds context beyond the objects or elements present in images. Kid-specific apps often use cartoonish characters, which are often associated with distinctive shapes and distinctive colour palettes and textures (e.g., bright colours, glitter texture, non-complex surface reflections), while dark tones (e.g. blood) or provocative lighting would be inappropriate for younger audiences, even if the content seems neutral [71].

In Figure 4.3, we provide some example cases of apps with a significant disparity between the content and style of their creatives relative to the app category and content rating. The app icon and screenshots for *The Puzzle Cakes* showcase cake related content and would have to rely on the style to associate it with a puzzle game. Conversely, *Pocket Love* and *Dirty Crown Scandal* employ a similar animated, cartoonish style but are aimed at distinctly different audiences based on their content ratings. Furthermore, there can be inter-app content and style disparities as in the examples of *The Virus* and *Spin the Bottle Game*. In both cases, the colour themes of the app icons are very different from the screenshots.

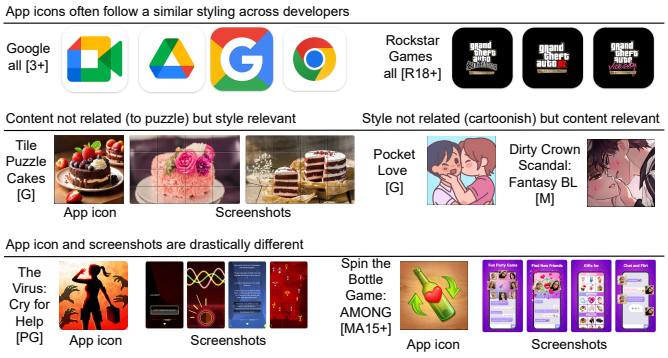


Figure 4.3: Disparity between the content and style of app icons and screenshots.

Therefore, we embedded a separate style encoder module. We use the CLIP image encoder as the base network for the style encoder, employing a masked representation learning task. This involves two identical networks, as illustrated in Figure4.2 (a). First, we uniformly sample an image i from the dataset D and generate an augmented image $x_s = t_s(i)$ by applying an image augmentation $t_s \sim T$. Next, we randomly mask three 3×3 patches to produce the masked image $x_s^{'}$. The masked image is provided for one network called *online*, while the unmasked image passes through another network called *target*, a slow-moving average network. This allows the network to focus on the features that are invariant to masking. The target network θ_t uses an exponential moving average (EMA) of the online network θ_s to learn lower semantic features [244, 113]. More precisely, given a target decay rate $\tau \in [0, 1]$, after each training step, we update the target network weights using,

$$\theta_t \leftarrow \tau\theta_t + (1 - \tau)\theta_s. \tag{4.1}$$

The EMA introduces stability by averaging the network's weights over time, smoothing out rapid updates that occur during the training process. This slows down the learning of fast-changing, higher-level features and enables the capturing of lower-level feature. From the masked image $x_s^{'}$, the online network outputs a representation $q_s^{'} = f_{\theta_s}(x_s^{'})$. The target network outputs $q_s = f_{\theta_t}(x_s)$ from the augmented view x_s. Finally we define the mean squared error between the embeddings q_s and $q_s^{'}$

$$\mathcal{L}_{mse} = ||q_s - q_s^{'}||_2^2. \tag{4.2}$$

and is added to the final loss, which we discuss later in Section 4.3.4. Furthermore, the generated style embeddings from the target network are scaled down using the hyperparameter α and added to the embeddings generated by the content backbone.

4.3.1.2 Content Encoder

As shown in Figure 4.2(a), our image content backbone, f_{θ_c}, is built on the CLIP image encoder and remains active throughout the training process. This branch operates on augmented images of 224x224 resolution, which could be derived from an app icon or screenshots. We follow the augmentation settings defined in CLIP to generate different views of the image. The content backbone outputs visual content features, q_c and are combined with visual style features, q_s given by the style encoder. We represent this combination using the equation $q_i = q_c + \alpha.q_s$ where α is empirically selected as 0.1.

4.3.2 Encoding Textual Information

The app description provides an overview of the app's functionalities, features, and content to the prospective audience. Often, the text is summarised as app users are reluctant to read lengthy texts (capped at 4,000 characters; the average length of an app description in the top 20,000 apps of Google Play Store is 2,169 words), and Google Play mandated it to be general audience friendly. We perform randomised text chunking with four or more consecutive sentences randomly extracted from the long app description to be paired with respective app visuals. We used a 110 million parameter RoBERTa text transformer with maximum 256 tokens marked as text backbone in Figure 4.2(a) to encode this information while the model parameters are kept frozen during the training.

4.3.3 Image-Text Cross Attention

Typical image captioning datasets such as MS-COCO [245] and Flickr30k [246] have a strong correlation between the captions and the images. In contrast, app icons/screenshots and descriptions can exhibit a larger semantic gap, especially in the context of the content rating prediction problem because: 1) the description may not perfectly reflect what is depicted in the app icon or screenshots, 2) these modalities may not always contain useful information related to the content/age rating, and 3) App images display complex variations within the same content rating class. Therefore, we employ a stack of cross attention layers [247] to align visual and textual tokens to fill the correlation gap between image patches and words. As shown in Figure 4.2(b), our custom cross-attention module initially has a self-attention layer followed by a cross-attention layer. This layer induces text information to the image features. The *query* values are derived from the previous image content encoder layer, and the memory *keys* and *values* are obtained from the hidden layers of the text encoder and vise versa. Text-to-image cross attention allows every patch of the image to attend over all tokens in the input sequence. Conversely, in *image-to-text cross attention*, the roles are reversed, allowing every token of the input sequence to attend over all patches in the content image. Finally, we introduce an additional linear projection layer, which outputs the final visual and textual embeddings.

4.3.4 Loss Function

Given a paired image and text sample (i, j) from dataset D, two transformations t_c and t_s are drawn from a distribution of image augmentation T (**cf.** 4.3.1), to produce two distinct views $x_c = t_c(i)$ and $x_s = t_s(i)$ of the image i. These views serve as inputs to the image content backbone f_{θ_c} and the image target encoder f_{θ_t} in the style encoder block, respectively. The views x_c and x_s are first encoded by f_{θ_c} and f_{θ_t} into their representations $q_c = f_{\theta_c}(x_c)$ and $q_s = f_{\theta_t}(x_s)$, which are then linearly combined to get the representation q_i. The text j is encoded

by a text backbone, g_ϕ into their representation $q_j = g_\phi(j)$. Then, these representations, q_j and q_i are mapped by the custom cross attention modules onto the embeddings z_i and z_j. The Sigmoid Contrastive Loss [142] is computed at the embedding level on z_i and z_j.

As defined in Equation 4.3, we adopt a supervised Contrastive Loss, more specifically, Sigmoid Contrastive loss (SigCL) proposed by [142] with content rating label, where we consider image-text pairs with the same rating as positive pairs. This enables it to distinguish between data points not just based on data similarity but also according to their categories. Additionally, the SigCL benefits over Unified Contrastive Loss (UniCL) [248] in a multi-modal setting because, when N image-text pairs from the same content rating category are presented in a batch, UniCL is bounded and the maximum softmax value per pair is limited to $1/N$. Meanwhile, SigCL varies between 0 and 1 for each positive pair. We defined the SigCL between z_i and z_j embeddings along the batch B as,

$$\mathcal{L}_{SigCL} = -\frac{1}{|P|}\sum_{i,j\in P} log\frac{1}{1+e^{(-\tau z_i.z_j+b)}} - \frac{1}{|B|}\sum_{i\in|B|}\sum_{j\in|B|\setminus\{P\}} log\frac{1}{1+e^{(\tau z_i.z_j-b)}} \quad (4.3)$$

where y_i and y_j are the labels for a given image and text pair and $P = \{k|k \in B, y_i = y_j\}$, which represents the image text pairs coming from the same content rating. The b in Equation 4.3 alleviates the heavy imbalance coming from the many negatives. We also employ Euclidean distance loss, L_{mse} to learn low level information such as texture and colour in image data. Specifically, we take the augmented image x_s and generate a masked version of it. The two views are encoded by the $target$ and $online$ networks described in Section 4.3.1.1 into representations y_s and y'_s, and optimise them using L_{mse}. The entire network is optimized by minimising the following loss function:

$$\mathcal{L} = \mathcal{L}_{SigCL} + \lambda\mathcal{L}_{mse} \quad (4.4)$$

where λ is a positive constant that controls the trade-off between the first and second terms in the loss $\mathcal{L}$.

4.3.5 Content Rating Classifier

For the downstream prediction task, we trained a classifier using the frozen image and text embeddings produced by the contrastive model described earlier. The content rating prediction task is inherently ordinal, where the classes follow $G \leq PG < M \leq MA15+ < R18+$. Therefore, we treat the task as an ordinal classification problem. Misclassifying an app into a lower rating category is more harmful than predicting a higher one. Therefore, the ranking used in our loss formulation is constructed to prefer the true label and any more restrictive (higher) ranking

labels over less restrictive (lower) ranking labels. This ensures that the classifier penalises unsafe under-prediction more strongly than over-prediction.

Let z_i and z_j denote the image and text embeddings for an app. After passing through two MLPs and concatenation, the classifier outputs a score vector $s \in \mathcal{R}^K$ over the $K = 5$ classes (**cf.** Figure 4.2(c)). Alongside the standard cross-entropy loss, we introduce a List-wise Maximum Likelihood Estimation (ListMLE) [56] term to explicitly encode the ordinal and safety-sensitive structure. ListMLE requires a ranking order π of each class. We construct π by prioritising the true class and progressively selecting classes in the direction that preserves user safety. Specifically, for an app with true label y, we use y as the rank-0. We then append the two higher-rated classes in increasing order, followed by the two lower-rated classes in decreasing order. This ordering reflects that predicting the true class or a higher rating is acceptable, whereas predicting a lower rating should be penalised more heavily. Formally, let the class index set $\mathcal{C} = \{1, \ldots, 5\}$ be ordered as [G, PG, M, MA15+, R18+].For a sample with true label y, we construct the ranking order π_y as $\pi_y = [\, y,\ y^{\uparrow},\ y^{\uparrow\uparrow},\ y^{\downarrow},\ y^{\downarrow\downarrow}\,]$, where $y^{\uparrow}$ and $y^{\uparrow\uparrow}$ denote the classes higher than y, and $y^{\downarrow}$ and $y^{\downarrow\downarrow}$ denote the classes lower than y, selected alternately until all classes are ranked. In Table 4.2, we show the selected class ranking matrix for all five content rating classes.

Given a ranking π, ListMLE models the probability of observing this ranking under scores s as,

$$P(\pi \mid s) = \prod_{t=1}^{K} \frac{\exp\left(s_{\pi_t}\right)}{\sum_{k=t}^{K} \exp\left(s_{\pi_k}\right)}, \tag{4.5}$$

and the corresponding ListMLE loss is given by

$$\mathcal{L}_{\text{ListMLE}} = -\log P(\pi \mid s) = -\sum_{t=1}^{K} \left[s_{\pi_t} - \log \sum_{k=t}^{K} \exp\left(s_{\pi_k}\right) \right]. \tag{4.6}$$

The final classifier objective combines cross-entropy with the ordinal regulariser:

$$\mathcal{L}_{\text{clf}} = \mathcal{L}_{\text{CE}} + \gamma \mathcal{L}_{\text{ListMLE}} \tag{4.7}$$

Equation 4.7 preserves the standard discriminative signal while encouraging the classifier to prefer higher-rank predictions under uncertainty, thereby reducing harmful underestimation errors.

4.4 Experimental Setup

4.4.1 Dataset

Our dataset is a snapshot of the Google Play Store, which includes metadata and creatives for 1.3 million apps. This dataset was collected using a Python crawler from January 2023 to November 2023. We deployed an extremely slow crawling rate during this data collection. For this work, we filtered out and used only the games category, which is more popular among children and as such, the correct content rating matters significantly. During our crawl, the crawler's geo-location was set as Australia (AU) to be consistent in obtaining content rating values of G, PG, M, MA15+, or R18+.

We sorted the selected gaming apps by rank, i.e., sorting in the descending order of number of downloads, star rating count and final star rating number following similar previous work [18, 21], and the first 20k games were selected as training and validation sets (80:20 random split) while the next 10k games were selected as the test set. We specifically did not mix the former due to the assumption that more popular apps are well monitored within the community and well maintained by the developers such that the metadata and content ratings information are less noisy than in the rest of the order. Due to the scarcity of games in categories of MA15+ and R18+ within the top 30k, we expanded our search space for them and appended them into train, validation and test sets. For analysis purposes, we created another dataset by including apps with the 'Teacher Approved' tag [226]. We report the distribution of apps by content rating across various datasets in Table 4.1.

Table 4.1: Dataset split and class distribution for train, validation, test and teacher-approved (TA) datasets.

	Train	Val.	Test	TA
G	4,544	1,139	2,650	2140
PG	4,540	1,130	2,649	30
M	4,530	1,120	2,648	2
MA15+	2,131	547	1,796	-
R18+	255	62	255	-

Table 4.2: Selected rank order for each class. Rank-0 represents the true class label.

Rank0	Rank1	Rank2	Rank3	Rank4
G	PG	M	MA15+	R18+
PG	M	MA15+	G	R18+
M	MA15+	R18+	PG	G
MA15+	R18+	M	PG	G
R18+	MA15+	M	PG	G

4.4.2 Implementation Details

We use the ViT-B/16 CLIP image encoder as our image backbone for both style and content branches and a frozen RoBERTa backbone in the text encoder branch. The model is pre-trained on eight NVIDIA V100 GPUs for 30 epochs with a minibatch size of 64. We use the Adam optimizer [249] with learning rate of 10^{-5}, momentum of 0.9, and weight decay of 0.02. The learning rate follows a cosine decay schedule [250], starting from 0 with 10 warmup epochs

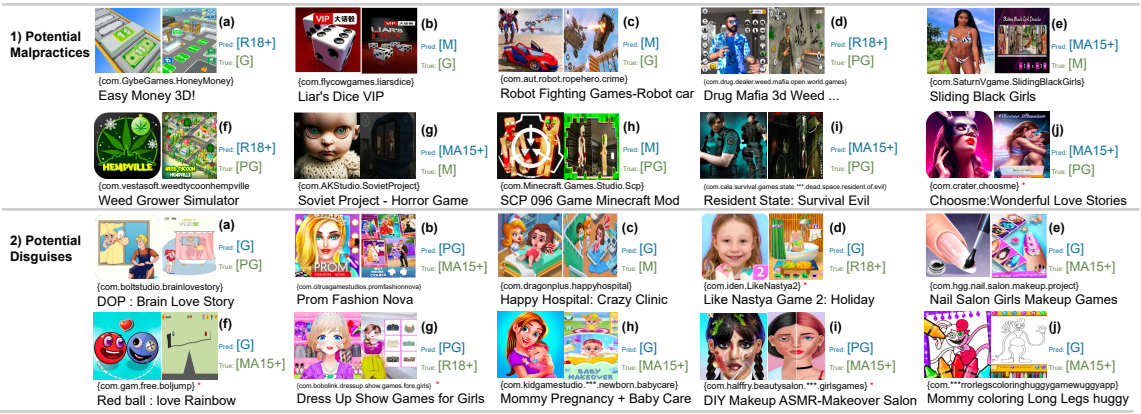


Figure 4.4: Examples belonging to 1) potential malpractices, and 2) potential disguises. For each app, the image on the left represents the app icon, and on the right is a screenshot. Red * represents app which removed from the Play Store after the initial data crawl in 2023.

and with a final value of 10^{-8}. We perform a grid search to select the loss coefficients λ in Equation 4.4 and set it to 5.

4.4.3 Performance Metrics

We evaluate the classifier using weighted and macro precision, recall, F1-score, and overall accuracy. In addition to these standard metrics, we introduce an Age-Safety Matrix (ASM), derived from the confusion matrix, to quantify the safety characteristics of misclassifications. Specifically, we report the proportion of samples that fall on the diagonal and in the upper-triangular region of this matrix. Predictions in the upper triangle correspond to assigning an app to its true rating or a higher content rating, which is a safer outcome for users. In contrast, lower triangular errors, where an app with a mature rating such as M or MA15+ is incorrectly predicted as suitable for younger audiences such as G or PG, pose a greater safety risk. The Age-Safety Matrix, therefore, captures both correctness and the directionality of errors, which is essential for evaluating models in safety-critical ordinal classification settings such as content rating prediction.

4.5 Results

In this section, we first compare the performance of our method against state-of-the-art baselines using image-only, text-only, and image–text multimodal embeddings. We then evaluate the proposed Age-Safety Matrix using the combined embeddings and analyse the classifier trained with cross-entropy alone versus cross-entropy with ListMLE regularisation. Next, we present an ablation study to assess the contribution of each model component by removing them individually. Finally, we examine the impact of alternative contrastive objectives by comparing

Table 4.3: Performance Evaluation. Linear classification on top of the frozen image and text representation and supervised baselines

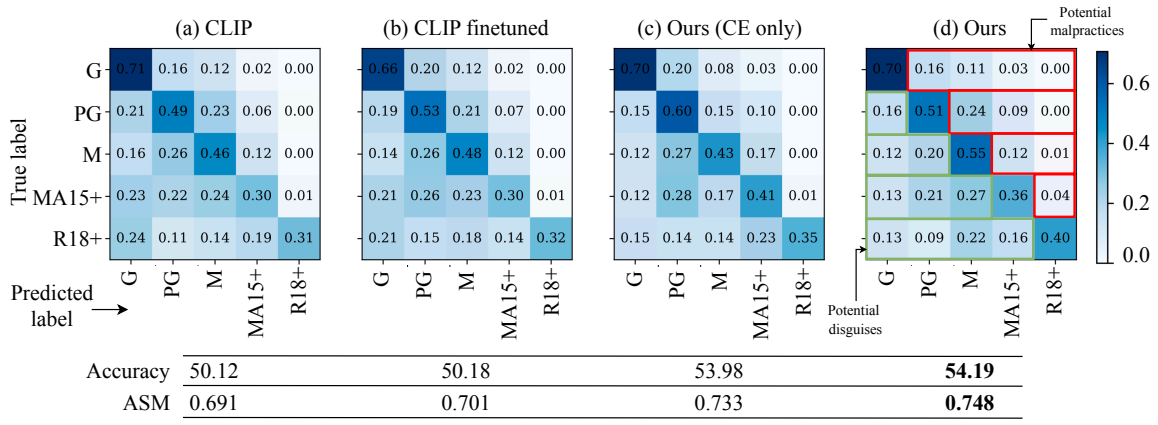
	P_m	R_m	P_w	R_w	Acc
1) Image embeddings only					
ResNet50	26.48	**45.54**	44.07	38.60	38.61
ViT	28.33	35.25	45.18	46.06	46.07
BLIP	40.48	32.74	48.65	45.18	45.19
CLIP	56.88	42.57	51.24	50.02	50.03
CLIP-f.t.	56.64	42.39	51.01	49.80	49.79
Ours	**61.18**	45.14	**53.10**	**51.14**	**51.15**
2) Text embeddings only					
BERT	45.28	**47.94**	51.06	49.32	49.33
RoBERTa	52.35	46.86	49.82	49.83	49.84
BLIP	38.10	41.20	45.11	39.75	39.76
CLIP	51.08	42.62	47.44	47.65	47.46
CLIP-f.t.	49.25	41.63	46.69	47.08	50.18
Ours	**60.58**	43.31	**52.54**	**50.45**	**50.46**
3) Image-Text embeddings only					
ViT+RoBERTa	33.33	33.87	48.45	50.78	50.78
BLIP	33.26	32.59	47.73	49.13	49.13
CLIP	53.66	45.44	49.97	50.11	50.12
CLIP-f.t.	**57.70**	45.56	50.60	50.19	50.18
Ours	53.01	**50.51**	**53.81**	**54.19**	**54.19**

symmetric cross-entropy (SCE) [50], UniCL [248], and SigCL [142] losses.

4.5.1 Performance Comparison with Baselines

Table 4.3 reports results across three evaluation settings: 1) image-only (z_i), 2) text-only (z_j), and 3) image–text (z_i and z_j) embeddings. We adapt the classification pipeline in Section 4.3.5 such that only z_i or z_j is passed through a single MLP before softmax prediction. In the image and image–text settings, predictions from all available app creatives (icon and screenshots) are aggregated via majority voting, whereas the text-only classifier uses a single description per app. For the CLIP baselines, we evaluate both pre-trained and fine-tuned variants: the pre-trained CLIP encoder is frozen before training the classification head, while the fine-tuned version is obtained by updating CLIP on our training set of app creatives and descriptions for 13 epochs (batch size 384), after which the encoder is frozen and only the classifier is trained.

We also evaluate standard supervised baselines trained end-to-end on our dataset. Image-only models use ResNet50 [251], ViT [206], and BLIP [127] image encoders; text-only models use BERT [112], RoBERTa [102], and BLIP text encoders; and multimodal baselines use concatenated ViT+RoBERTa and BLIP image–text encoders. These baselines illustrate the limitations of relying solely on conventional image or text towers. ResNet50 and ViT show large drops in accuracy compared to our image-only classifier (-32.48% and -11.03%), reflecting



	(a) CLIP	(b) CLIP finetuned	(c) Ours (CE only)	(d) Ours
Accuracy	50.12	50.18	53.98	**54.19**
ASM	0.691	0.701	0.733	**0.748**

Figure 4.5: Confusion matrices comparing our method against baselines - using image-text embeddings.

the high visual variability across app creatives. In contrast, BERT and RoBERTa experience a smaller average decrease ($\sim$1.8%), as app descriptions often contain explicit age-relevant information. Nonetheless, our approach using only the text embedding surpasses these standard text baselines due to the contrastive pre-training and cross-modal alignment that enrich both text and image embeddings. Similarly, the ViT+RoBERTa and BLIP multimodal baselines achieve 50.78% and 49.13% accuracy respectively, underperforming our method by 6.72% and 10.3%.

When using both image and text embeddings, our model attains 54.19% accuracy, yielding relative gains of 8.12% over pre-trained CLIP and 7.99% over CLIP-ft. Similar improvements are observed for macro and weighted precision and recall. Although the image-only and text-only variants exhibit lower absolute accuracy (-5.94% and -7.39% relative to the multimodal setting), they still surpass all CLIP and CLIP-ft baselines across every metric excluding R_m . These results indicate that the combination of the style encoder and image–text cross-attention produces richer representations than those obtained from independent image or text towers, and leads to consistent improvements across all evaluation regimes.

4.5.2 Age Safety Evaluation

Content rating categories differ in qualitative, regulatory, and culturally defined ways. The semantic gap between MA15+ and R18+ (e.g., graphic violence, sexual content, extreme themes) is not comparable to that between G and PG (e.g., mild suspense or language). This motivates the use of an ordinal order-based regulariser rather than an ordinal distance-based regulariser. Therefore, we adopt ListMLE to impose a ranking structure over class logits. As shown in Equation 4.7, cross-entropy encourages prediction of the correct label, while ListMLE turns the model's logits into a monotonic severity scale. This enables the model to encode a continuous notion of severity across classes. As illustrated in Table 4.2, the ranking order is constructed to

Table 4.4: Performance with different losses

	Metric	SCE	UniCL	Ours
Macro	Precision	56.36	**60.29**	53.01
	Recall	38.52	43.45	**50.51**
	F1 Score	38.17	45.23	**51.39**
Weighted	Precision	49.74	52.63	**53.81**
	Recall	48.59	51.51	**54.19**
	F1 Score	46.77	50.50	**53.75**
	Accuracy	48.59	51.52	**54.19**

prioritise child safety by placing lower-severity ratings far from higher-severity ones. For example, a sample with true label M is assigned the ranking [M, MA15+, R18+, PG, G], ensuring that its representation remains closer to adjacent higher-severity classes than to substantially milder ones.

To quantify safety-oriented behaviour, we introduce the Age-Safety Matrix (ASM), defined as the proportion of samples that fall within the upper triangular region of the confusion matrix, where predictions are equal to or stricter than the ground-truth rating. As shown in Figure 4.5, CLIP and CLIP-ft exhibit high accuracy for lower-risk classes (G, PG, M) but reduced accuracy for higher-risk categories (MA15+, R18+). A similar pattern is observed in our model when trained with cross-entropy alone. Incorporating ListMLE substantially improves safety alignment: predictions become more uniformly distributed across all classes, and a larger fraction of samples shift into the upper-triangular region. Our method achieves an ASM of 0.748, meaning 74.8% of predictions are at or above the true rating, compared to 0.691, 0.701, and 0.733 for CLIP, CLIP-ft, and our cross-entropy-only variant, respectively.

4.5.3 Ablation Studies

Ablations with respect to loss functions: We alter our contrastive loss function in Equation 4.3 in several ways to experiment with different loss functions in SSL, such as SCE loss, UniCL, and SigCL. We compare linear classification results trained on the combination of image and text representations. The results of Table 4.4 show that our method trained on SigCL outperforms SCE and UniCL methods by 11.53% and 5.18%, respectively.

Ablation of model components: To observe the effect of the style encoder, we evaluate our model with only the content-encoder as presented in Table 4.5. The macro average precision observes a gain of 18.1% without the style encoder (i.e., less likely to make false predictions but recalls less: -15.96%), yet, all the other metrics indicate better performance with our method (macro average F1 score: +12.10%, accuracy: +6.82%). Next, we augmented our methodology without cross-attention and replaced it with a self-attention block. Our method achieves better

Table 4.5: Effect of incorporating style encoder and cross attention

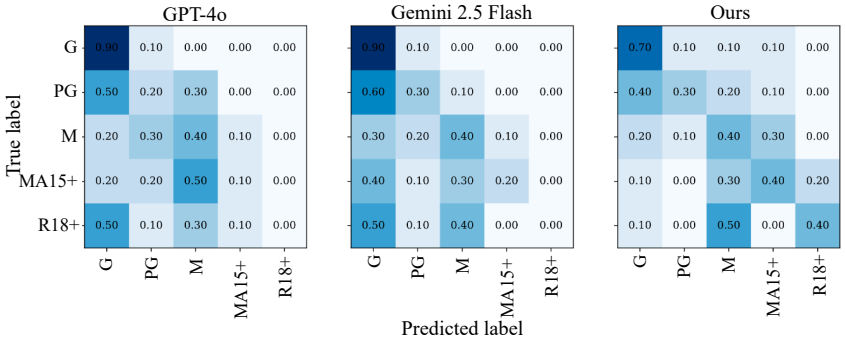
	Metric	w/o style encoder	w/o cross attention	Ours
Macro	Precision	**62.60**	58.81	53.01
	Recall	43.56	46.43	**50.51**
	F1 Score	45.84	49.12	**51.39**
Weighted	Precision	**54.91**	50.99	53.81
	Recall	50.72	50.76	**54.19**
	F1 Score	49.86	49.80	**53.75**
	Accuracy	50.73	50.76	**54.19**

performance in all the metrics compared to this setting. Removing cross-attention disproportionately affects classes, as indicated by a larger drop in the weighted F1 score compared to the macro F1 score. This suggests that cross-attention is crucial for maintaining performance in majority classes like G, PG, and M, helping the model effectively distinguish between these content ratings. Overall, ablation study results show that style backbone added with text-image cross-attention contributes to the increased performance.

GPT-4o vs. Our Method: To further evaluate the performance of our proposed model, we conducted an ablation study comparing it with GPT-4o and Gemni 2.5-Flash models. For this experiment, we selected the 50 games from our test set, with top 10 samples per class. We formulated a prompt combining the app description, the image, and content rating definitions, querying the model for its predicted content rating. Figure 4.6 presents the confusion matrices for all three models. While GPT-4o and Gemini 2.5- Flash demonstrate an overall accuracy of 32% and 36% respectively, our method significantly outperforms it, achieving 44.00% accuracy, an 37.5% and 22.2% improvement. In particular, both LMMs fail to correctly predict any instance of the R18+ class, highlighting its limitations in identifying mature content. In contrast, our model successfully identifies multiple samples from both MA15+ and R18+ classes and demonstrates improved precision and recall across all categories. Interestingly, both LMMs tend to under predict the content rating (lower ASM), often assigning lower classifications than appropriate, particularly for mature content. This tendency may reflect a bias toward conservative or safe outputs in the absence of explicit domain tuning. These findings highlight the advantages of our task-specific model in capturing the nuanced cues required for accurate content rating prediction.

4.6 Result Analysis

In this section, we analyse the results and predictions of our method from the perspective of the mobile app ecosystem. First, we observe how deviated the text and visual data are compared to natural language and text. Next, we present interesting findings among the lower and upper triangular parts of the confusion matrix (i.e., apps having a lower or higher content rating than



	Precision	Recall	Accuracy	ASM
GPT-4o	24.27	32.00	32.00	0.42
Gemini 2.5-Flash	34.17	36.00	36.00	0.42
Ours	**49.89**	**44.00**	**44.00**	**0.68**

Figure 4.6: Confusion matrices comparing our method, GPT-4o and Gemini 2.5 flash.

our predictions).

4.6.1 Image-text Cross Attention

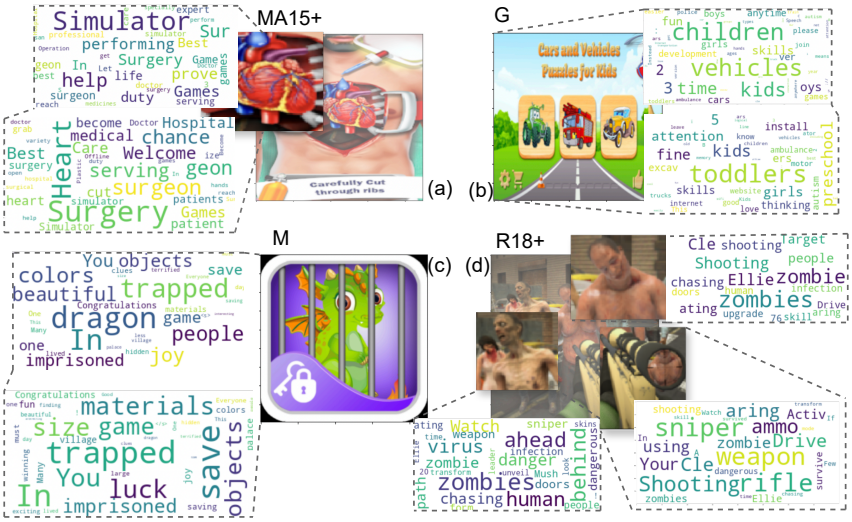


Figure 4.7: Visualisation of image patches attending to text tokens in the custom cross-attention layer.

In Section 4.3.3, we discussed how our image-text cross attention (CA) design allows every patch of the content image to attend over all tokens in the text input sequence. Also, for each image patch, there are 12 attention heads running in parallel. Therefore, to qualitatively measure how the CA happens, we select attention-heads of the first layer as lower layers are often associated with broader attention [252]. Then, for each image patch, we select the tokens with the highest numerical attention values after excluding some stop-word and punctuation mark-related tokens. In Figure 4.7, we visualise the highly attended words in a given input image portion for some example images. An image portion is a collection of consecutive patches which we select as a region of interest; for example, the patches outlining the gun in Figure 4.7

(d) or the heart in Figure 4.7 (a). The results show that mobile app ecosystem-related tokens such as 'simulator', 'game', 'upgrade', and 'developers' are now attended by the image patches, even though those words cannot be identified by observing the image in a general context. Furthermore, the tokens representing target audiences such as 'kids', 'children' and 'girls' are now attended. This further demonstrates how our model has been able to reduce the gap between visual data and textual data in the mobile app domain.

4.6.2 Predictions in the Wild

When our classifier is applied to the test set containing 10,000 gaming apps, two interesting cases emerge, i.e., apps with a higher or lower labelled rating than our model's predictions. These two scenarios lead to potential *"malpractices"* and *"disguises"* that are important from a content safety point of view.

4.6.2.1 Potential malpractices

When our method predicts a class label that is higher than the developer-defined label, we characterise such examples as possible malpractices (i.e., having a lower content rating than what the app is supposed to have - cf. Figure 4.5 (d)). Occurrences belonging to this category could be identified along the horizontal axis of the confusion matrix, on the right-hand side to the diagonal. This is more observed in the G category as any higher prediction, such as PG, M, MA15+, and R18+, raises a concern. As we go higher in prediction classes, the possibilities for malpractices decrease, and therefore, the R18+ category is not susceptible to malpractices.

We manually evaluated 350 apps with predicted labels that were two classes or more higher than the true label (e.g., prediction [M, MA15+ or R18+] when the true label is [G]) and identified 62 (17.7%) of them as potential malpractices (i.e., possible content rating violations). *Also, we highlight that at the time of writing, 14 of them were removed from Play Store, and 20 of them were increased to a higher rating class compared to the time we crawled the dataset. While we can't be exactly sure why these 14 apps were removed from Google Play Store, previous work has reported that Google take down apps violating their content policies [18].*

We highlight ten exemplary instances that are potentially linked to malpractices in Figure 4.4 (1). Examples (a) and (b) both represent gambling-related games rated [G] that would at least require a rating of [PG]. Example (c) depicts shooting and gun usage, again not suitable for a general audience. (e) and (j) contain images more suited for an [MA15+] audience, and (d) and (f) entertain visual and textual cues strongly related to illegal substances that require a rating higher than [PG]. (g) and (h) contain images with horror themes, blood and intense cartoon or fantasy violence which are more suited for [MA15+] audience. These examples suggest that our model can flag potential content malpractices in the app ecosystem.

4.6.2.2 Potential disguises

Apps belonging to higher content ratings are likely not to conduct malpractices, but alarmingly, they could be disguised as a lower-rated app and, as a consequence, could attract an unsuitable audience to the app (cf. Figure 4.5 (d)). As an example, an R18+ app consisting of cartoonish images and not-so-alarming textual data could attract underage audiences due to their natural tendency for curiosity and their interest in cartoons. From the developer's perspective, they comply with the content policies and applicable laws. However, in such cases, we argue that at least the textual description must contain information such that someone (e.g., a parent or a guardian) who misses the content rating label should be able to figure out the app's purpose and functionality independently. We define this category as possible disguises. R18+ is more vulnerable to disguises that can be identified horizontally left to the diagonal of the confusion matrix. Being the lowest content rating, category [G] is not susceptible to disguises. We checked 131 samples that were predicted [G, PG, M] when the true class was [R18+], and 203 samples predicted [G] when the true class was [MA15+] and observed $\sim$9.3% of them to be possible disguises.

We demonstrate some of such examples in Figure 4.4 (2). App represented by (a) is an app rated for [PG] and our method predicts even lower as [G]. Observing the images, it is evident their content is sexual in nature despite the cartoonish theme. In our test dataset, the average rating for an app with such sexualised imagery is [MA15+]. Note that in this category, our model is likely to predict a lower rating as the images or text is not suggestive of requiring high ratings. Due to such examples being rare, our model does not know how to predict them correctly, yet we still can automatically identify them based on our off-to-left-diagonal results, as explained before. Apps (b, e, g) and (i), though rated for [MA15+] and higher based on a storyline related to a 'fashion and makeup', is likely to attract a younger audience due to the visual appearance similar to the majority of [PG] rated *Dress up* games. A similar interpretation can be given to [M] rated examples (c, h), which are too cartoonish yet contain images related to mature audiences, such as pregnancy. Furthermore, (f) and (j) are rated as [MA15+], which contains appealing games for young audience. Despite measures such as *Not designed for children* tagging is available in Google Play [253] to safeguard users, it doesn't appear to be used by many developers.

4.6.2.3 Unverifiable apps

We further highlight discrepancies in content rating declarations, as illustrated in Figure 4.8. These noisy instances represent cases where we could not manually verify the developer-assigned ratings based on the app's available metadata, visuals, or descriptions. The app in (a) is rated

Figure 4.8: Examples of unverifiable apps with developer defined content rating descriptors.

as [M] with descriptors indicating simulated gambling, online interactivity, and in-game purchases. However, compared to similar games in the dataset, it does not display any visual cues or textual descriptions to justify such a rating. Therefore, the absence of gambling graphics or explicit mention of gambling mechanics justifies the predicted rating of [G]. Similarly, app (b) and (c) fail to reflect sensitive content such as sexualised imagery and strong violence in app creatives. Hence, in all these cases, our model predicted a lower rating than the original rating. While it may not be explicitly illegal to omit sensitive descriptors from app screenshots or descriptions, missing descriptors in visuals hinder the user's ability to make informed decisions, leading to unwanted downloads or unexpected experiences and children and vulnerable users may unintentionally be exposed to harmful or age-inappropriate content.

4.6.2.4 Teacher Approved (TA) Apps

Google Play has deployed the 'teacher approved' apps based on consultations with experts to determine the suitability for kids, and especially the age appropriateness [226]. Hence, these apps are likely to be more content appropriate as per the rating labelling. However, after analysing 2,172 TA apps, we found that our model predictions deviate from the declared content rating classifications, with 10.4% of them being flagged for potential malpractices. Among them, 92% of the instances were flagged as requiring [PG] despite being declared as [G]. Further evaluating their continuity, within a time span of nine months, we observed that 34.5% of apps that were classified as potential malpractices have been removed from the Play Store. On the contrary, only 27.4% of correctly predicted apps were removed, which is lower compared to the deletion rate of apps identified with malpractices. We further discuss why app removal rate can be a

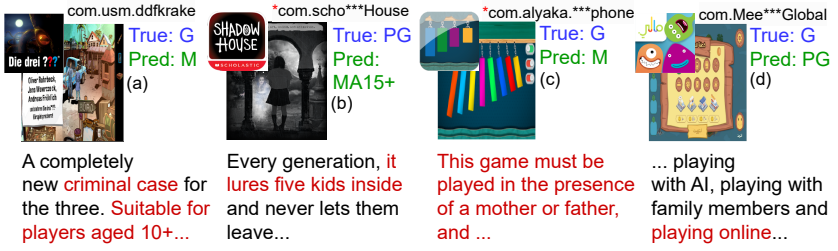


Figure 4.9: Examples of teacher-approved apps with incorrect content ratings - A section of the app description is quoted and * indicates this app is no longer available.

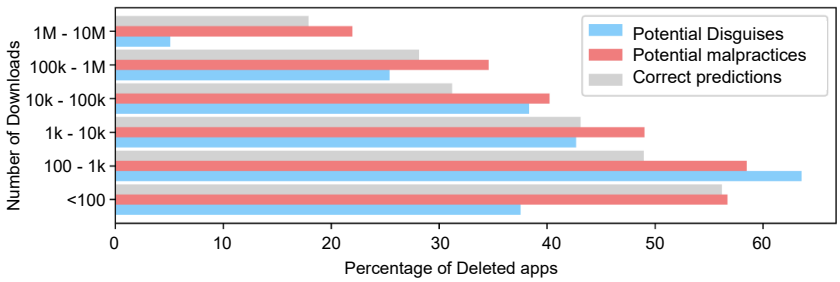


Figure 4.10: App deletion rates w.r.t number of downloads.

proxy measure of content policy violations in the next subsection.

As depicted in Figure 4.9 we manually verified a portion of apps flagged before as malpractices based on the available metadata and identified that nine apps are likely to be not suitable for children despite being tagged as TA. The presence of violence (Figure 4.9a), horror themes (Figure 4.9b) or online multiplayer interactions (Figure 4.9d) were the main reasons we identified behind these content rating discrepancies. The example in Figure 4.9 c highlights a contradiction in the app's description, which mandates parental presence, despite the app being labeled as [G] in the Play Store. *Overall, the presence of these practices among 'teacher approved' apps is alarming. It shows that even manually verified apps are not immune to content rating malpractices, and further rigour is required in app vetting.*

4.6.2.5 App Deletion Rate

An app can be discontinued in Google Play for two reasons: the developer could discontinue the app [254], or Google could remove the app for violating its policies [255]. As a result, an app being removed from Google Play can be used as an indication of a possible violation of Google Play policies.

To this end, we used 15,985 apps that are gathered from the test set (10,000 apps: c.f. Section 4.4.1) added with 5,985 apps with lower downloads (download count $<$ 100,000 - to account for a better distribution as our test set consists of top apps only). Next, we attempted to re-crawl these apps to check whether they were still there in Google Play. Overall, we found 45.7% of apps identified as having malpractices, 39.1% of apps that predicted to be disguises were removed within the time span of nine months. In comparison only 29.1% of apps that correctly predicted were removed.

In Figure 4.10, we show the percentage of apps that we found as deleted according to the download numbers and the predictions of our classifier. At all download ranges apart from $<$ 100, we notice that apps we classified as potential malpractices have a higher deletion rate than apps we classified as correct. Similar values of $<$ 100 category can be explained by less attention and consequently fewer complaints received on those apps for Google to action.

On the other hand, apps classified as disguises are likely not to be removed as malpractices,

as they are unlikely to be noticed or complained about by an average audience. Notably, apps flagged as disguises with more than 1M downloads are far less likely to be removed as the number of apps with a higher content rating label (e.g., MA15+, R18+) are not frequent among the apps.

4.7 Concluding Remarks

In this chapter, we proposed a vision-language approach to detect content rating violations in the Google Play Store. Our model includes multiple trained encoders capturing features related to app creative styles, content, text descriptions and their relationships using a cross-attention module. We employed ListMLE loss as a regulariser to predict the ordinal content rating prediction task. We trained our model using a large dataset from the Google Play Store focusing on gaming apps. Our method outperformed the state-of-the-art CLIP model, even when fine-tuned on the same dataset. We achieved 8.12% and 7.99% relative improvements in accuracy compared to CLIP and fine-tuned CLIP, respectively. Even though our method doesn't achieve perfect accuracy, apps that deviate from the predicted rating (i.e., potential malpractices or disguises) can serve as a shortlist for e-safety regulators and app market operators, thereby reducing manual effort. By leveraging static information such as images and text, we can quickly identify apps for further inspection. Beyond these two categories, we also identified unverifiable apps that have been assigned higher content ratings, yet even human reviewers could not justify these ratings based on the app creatives and descriptions alone. While there is no legal requirement for an app's creatives to explicitly reflect its content rating, we emphasize the importance of ensuring alignment between content descriptors and app creatives/descriptions. This transparency will help users to make informed decisions.

We applied our model in the wild and found that our model can detect content rating malpractices in practice. We could identify 71 ($\sim$17% of what we verified) of such examples. Some notable examples include gambling apps such as *Liar's Dice VIP* being categorised as G and *Drug Mafia 3d Weed* being categorised as PG. In addition to that, within our test set, 16.86% of the apps we identified as violating content policies are no longer available in Google Play Store due to potential banning, further justifying the effectiveness of our method. As an artefact, we also found another interesting behaviour related to app content ratings in the Google Play Store characterised as *potentially disguises*. These apps have a correct content rating. However, their look and feel appear to target a general audience. For example, an app with a cartoonish theme but mature content may inadvertently attract children, for whom the content could be disturbing. We identified 32 such instances. Finally, we conducted an extended evaluation on 2,172 'teacher approved' apps and identified nine apps with possible content rating malpractices.

One limitation of our work is the reliance on top apps having reliable content ratings and

representative app creatives. While several comparable works have used similar ideas in domains such as spam app detection [18], counterfeit detection [22], this approach may introduce noise into the CLIP fine-tuning process. One way to mitigate this limitation is through human annotation, though this can be costly. Another approach is to match apps between the Google Play Store and Apple App Store using a method such as [256], leveraging Apple's content ratings, which typically undergo manual verification. However, Google and Apple use different content rating scales, which may introduce inconsistencies.

Additionally, incorporating other app metadata—such as user comments, data safety declarations, and dynamic app behaviors—could enhance the framework's robustness against unverifiable apps. However, this approach is more resource- and time-intensive than analyzing text descriptions and app creatives. As a result, it could serve as a secondary classifier after identifying potential content rating violations at scale using our proposed method.

Chapter 5

QwenSafe: Multimodal Content Rating Descriptor Identification via Preference-Aligned VLMs

In the previous chapter, we leveraged the mobile app metadata to predict the content rating itself via self self-supervised contrastive learning approach. However, this method lacks a reasoning aspect. Therefore, in this chapter[1], we build upon content rating descriptors to enable fine-grained detection of inappropriate content in mobile applications.

5.1 Introduction

Mobile applications have become an integral part of everyday life, used not only by adults but also by children. With an average of 1,863 new apps being added to Apple's App Store [4] and 1,617 to Google Play [3] every single day, users are continuously exposed to novel experiences and functionalities. Given this widespread usage, ensuring that applications provide age-appropriate and safe content is a critical responsibility for app distribution platforms. However, not all apps adhere to established policies, and in a profit-driven ecosystem, developers may sometimes attempt to circumvent platform rules, resulting in apps that expose users, especially children, to harmful or inappropriate content through inaccurate age or content ratings. These issues undermine the integrity of the mobile app ecosystem, emphasizing the need for more robust and effective content rating systems.

The two leading app marketplaces, Google Play Store and Apple App Store, have implemented multiple safeguards to protect users, including strict developer policies, inspection and

[1]This chapter includes the work in: **D. Denipitiyage**, S. Seneviratne, and A. Seneviratne, "QwenSafe: Multimodal Content Rating Descriptor Identification via Preference-Aligned VLMs", ACM ASIA Conference on Computer and Communications Security (ACM ASIACCS) 2026 - Under review.

vetting procedures prior to publication, the provision of essential app information (e.g., download counts, permissions, ratings, developer details, and community reviews), and the assignment of age-appropriateness ratings to guide user decision-making. Android follows a region-specific rating schemes (see Section 1 and Figure 1.2). In contrast, Apple employs a centralised rating framework across all regions, along with regional exceptions. These rating schemes assess the presence of content descriptors such as mature themes, sexuality or nudity, violence, gambling, drugs and language [257, 37]. The initial evaluation is based on questionnaires completed by app developers when submitting their apps, with each store using its own questionnaire. In 2017, Google employed text analysis, image understanding, and static and dynamic analysis of the APK binary to identify potential threats an app poses to its users [258]. In 2023, Google introduced machine learning based initial screening to scan apps for policy violations, which reduces the manual workload and filters bad actors before they are ever seen by human reviewers [259, 260].

Despite Google's efforts to prevent the publication of apps that violate content rating schemes, a significant 19.25% of content rating inconsistencies were still observed across different agencies [47]. Additionally, 81.25% of apps categorised as "family-friendly" [261] were found to embed trackers, despite such practices being prohibited for children's apps [47]. The Canadian Centre for Child Protection (C3P) found that Google Play often assigns lower age ratings (e.g., "Teen") than the Apple App Store (e.g., "17+"), allowing younger users access to apps such as YouTube, KIK, Whisper™, and Yubo. They also note that Google Play provides less detailed content descriptors, limiting its descriptors to interactive elements and in-app purchases, while Apple includes more comprehensive content descriptors [46]. Furthermore, the C3P states that Apple uses content descriptors similar to the ESRB [41]. However, the downside is, manual review is resource-intensive and may substantially delay the release of an app to users.

Therefore, there is a pressing need to develop a centralised system that can be applied across all app marketplaces to address inconsistencies in age rating standards. Such a system would promote greater consistency and transparency in content classification, reduce confusion for developers and consumers, and strengthen child protection by ensuring that applications are evaluated against uniform criteria regardless of platform. In this chapter, we introduce an automated framework for analysing content rating descriptors (CRDs) directly from multimodal app metadata. We construct a unified descriptor taxonomy and generate descriptor-aligned training data that integrates app descriptions, screenshots, and formal descriptor definitions. We then adapt the Qwen3-VL-8B model [156], referred to as *QwenSafe*, for fine-grained CRD identification. QwenSafe consists of two components: (1) *metadata2CRD*, which converts visual and textual metadata into structured CRD question–answer pairs, and (2) an offline model-adaptation stage where supervised fine-tuning and Direct Preference Optimisation (DPO) [262] are used to align the model with descriptor-specific reasoning. Our contributions are as follows:

- We develop **QwenSafe**, a vision–language model adapted for CRD identification, supported by **metadata2CRD**, a data construction pipeline that produces descriptor-aligned Q&A pairs by combining app descriptions, screenshots, and formal descriptor definitions.
- We apply **supervised fine-tuning** and **DPO-based preference alignment** to Qwen3-VL-8B, enabling the model to identify descriptor presence and explain relevant cues across visual and textual modalities.
- Through evaluation on 12 Apple-defined descriptors, QwenSafe outperforms Qwen3-VL, LLaVA-1.6, and Gemini-2.5-Flash, improving positive class recall by **111.8%**, **36.1%**, and **2.1%**, respectively in binary CRD classification. In multi-class classification, QwenSafe increases average mild-class recall by **62.75%** over Gemini-2.5-Flash and achieves the highest strong-class recall, exceeding Gemini-2.5-Flash by **12.64%**.

5.2 Background

5.2.1 Legal Background and App Market Policies

Legislation: This section describes the existing legal/regulatory implementations such as the European General Data Protection Regulation (GDPR), the Children's Online Privacy Protection Act (COPPA), App Store and Google Play Policies around mobile app content rating, age assurance, and protecting minors from harmful content. Further, we specifically outline recent changes in mobile app content rating in Australia, as this research was conducted by setting the geographical location as Australia.

Children need constant vigilance and effort to protect their personal data, as they often do not fully understand the risks of how their data is collected and used. According to COPPA §312.4 [30], applications directed at users under the age of 13 are required to make reasonable efforts to ensure that a child's parent receives direct notice of the app's practices concerning the collection, use, or disclosure of personal information. Similarly, under GDPR [28] Art. 8, the processing of a child's (below the age of 16 years) personal data is lawful only when consent is provided by the holder of parental responsibility and Art. 5 requires that user personal data be processed "lawfully, fairly and in a transparent manner in relation to the data subject." These regulatory efforts collectively protect the underage audience from intentional or accidental access to age-restricted applications. While reputed organisations strictly follow such regulatory guidelines and reflect the target audience via content rating labels, there is no guarantee that not-on-spotlight developers are representing the actual content of their apps via content-rating labels [263]. Enforcement actions for such apps, therefore, are reactive in nature, as the regulatory efforts are predominantly focused on restricting access and securing personal information handling of children rather than verifying the content's suitability.

In Australia, the Online Safety Act 2021 (Cth), Part 9 [264, 265] consolidated powers of the eSafety Commissioner to compel removal and restrict access of harmful mobile apps, complementing the Broadcasting Services Act 1992 (Cth) [266] and its classification regime. The subsequent Online Safety Amendment (Social Media Minimum Age) Act 2024 introduced Part 4A, establishing a statutory minimum age of sixteen for users of designated social media platforms [265]. By requiring providers to take reasonable steps to prevent underage users from creating or maintaining accounts, and to purge existing accounts belonging to children, the amendment seeks to reduce children's exposure to harmful or age-inappropriate content. This law is intended to apply to companies such as Facebook rather than companies offering education and health support.

Content descriptors: Content descriptors are indicators accompanying age ratings that specify the nature of potentially sensitive or objectionable material present in an app. These indicators are defined by a regulatory body or by the app market itself. For example, ACB guidelines use a hierarchical descriptor scheme where the first layer has eight different indicators, themes, violence, sex, language, drug use, nudity, online interactivity, and in-game purchases, and assess the impact to decide the age rating. More specifically, Android follows five Likert-style scales: 'very mild', 'mild', 'available' and 'strong' and 'very strong'. iOS apps follow a similar structure containing seven main descriptors (In-App Controls, Capabilities, Mature Themes, Medical or Wellness, Sexuality or Nudity, Violence, Horror/Fear Themes, Chance-Based Activities) and consider 'mild/infrequent' or 'intense/frequent' presence of the descriptor. Even though both app markets have different first layers in their hierarchy, they both follows almost similar descriptors in the second layer. However, ACB provides a fine granular second layer compared to Apple. For example ACB divides violence into 12 sub-categories whereas Apple has only four sub-categories. Compared to Android, Apple recently introduces Parental Controls and Age Assurance as a protection layer for children under 16 years.

5.2.2 iOS and Android Content Rating Schemes

Content ratings and descriptors can diverge substantially even for the same application when distributed across iOS and Android, owing to the fact that developers respond to different platform-specific questionnaires during the submission process. As illustrated in Figure 5.1, $The\ Sims^{TM}$ ***FreePlay*** by ***Electronic Arts*** receives an R18+ rating on Android but only 15+ on iOS, despite representing the same title with identical screenshots. The underlying descriptors further highlight this discrepancy: Android attributes its classification to simulated gambling and in-game purchases, whereas iOS cites sexual content and nudity, alcohol and tobacco use, mature or suggestive themes, and several additional factors. Such inconsistencies raise a fundamental question about how two rating schemes can govern an identical game within the same region and which set of descriptors more accurately reflect the app's content. This phenomenon

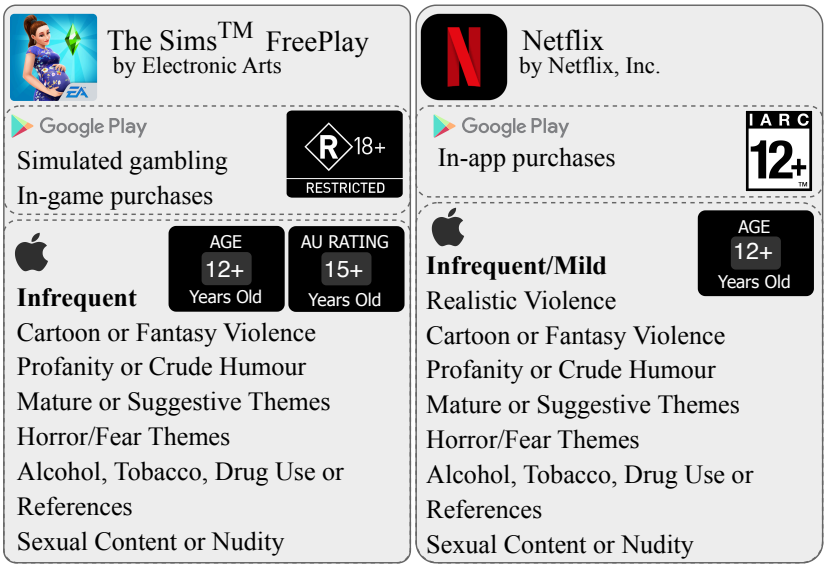


Figure 5.1: Content descriptors of $The\ Sims^{TM}\ FreePlay$ and $Netflix$ apps across App Store and Play Store.

extends beyond games. For instance, *Netflix* is rated 12+ on both platforms, yet the rationale differs markedly: Android points solely to in-app purchases, while iOS lists realistic violence, profanity or crude humour, mature or suggestive themes, sexual content and nudity, and several others. More concerning is that developers appear to treat each app marketplace as a distinct regulatory environment, even when distributing the same product. In practice, households often operate a mix of Android and iOS devices, leaving end users, particularly parents, uncertain about which rating or set of descriptors to trust when evaluating apps they have no prior familiarity with.

Additionally, we observed several content rating descriptors that are visible only in certain app ecosystems. In iOS, if developers claim their game contains infrequent nudity or sexual content, that app may no longer receive a content rating less than 12+. In Android, very mild nudity or sexual content could be present in a general (G) rated game, and more occurrences require parental guidance. It is again concerning how Australian regulators are not binding both behaviours to avoid this confusion to both app developers and end-users alike. In this work, we aim to reduce this gap by comparing both ecosystems.

5.3 Related Work

Content Rating: Comparatively fewer efforts directly address the problem of automatic app maturity rating, despite increasing concerns regarding inappropriate content exposure and misclassification in app marketplaces. Early work by Chen et al. [53] proposed Automatic Label of Maturity ratings (ALM), a semi-supervised text-mining approach that infers maturity ratings

using app descriptions and user reviews, treating Apple App Store ratings as ground truth. However, this method relies primarily on keyword matching and lacks semantic understanding. Hu et al. [55] extended this line of work by introducing a text feature–based SVM classifier with online training, but similarly depended exclusively on textual cues. Subsequent studies by Liu et al. [71] and Chenyu et al. [54] incorporated additional modalities, including app icons, screenshots, and APK features, to identify children's apps. Nevertheless, their feature extraction was limited to surface-level signals such as OCR-extracted text, colour distributions, permissions, and APIs, without deeper cross-modal reasoning.

More recent work by Sun et al. [47] analysed inconsistencies in content ratings across geographic regions by mapping rating systems between jurisdictions. Complementing these findings, an investigation by the Canadian Centre for Child Protection Inc. (C3P) [46] in 2022 reported that app age ratings often vary across Apple's App Store, Google Play, and the app's own terms of service, and further noted that both major mobile app stores lack transparency in how age ratings are determined. Beyond automated methods, longitudinal audits have revealed substantial non-compliance with platform and regulatory standards. For example, Xiao [267] reported widespread violations in the UK loot box industry, where apps rated as suitable for children (e.g., 4+ or 9+) nonetheless imposed 18+ age requirements through in-app gating mechanisms. Similarly, Carter et al. [268] identified systemic failures in Australia's Mandatory Minimum Classifications Scheme for gambling-like content. Collectively, these studies underscore persistent gaps between declared ratings and actual content exposure, motivating automated, content-descriptor-aware auditing tools.

Generative AI models for harmful content understanding: Recent years have seen growing interest in leveraging large language models (LLMs) and vision–language models (VLMs) for detecting inappropriate, hateful, or otherwise harmful content across online platforms. Early work [269] evaluated the ability of GPT-3 to identify sexist and racist text, demonstrating that few-shot prompting could achieve accuracies of approximately 85%. Building on this line of research, Guo et al. [270] systematically explored four prompting strategies for online hate speech detection using LLMs, reporting substantial F1-score improvements ranging from 7.9% to 24.2% on the HateXplain dataset [271]. Beyond text-only approaches, VLMs extend content understanding by jointly modelling visual and textual signals, enabling the detection of multimodal harmful content. The Hateful Memes Challenge [169] introduced a benchmark highlighting the limitations of unimodal models and the necessity of multimodal reasoning. ProCap [170] employed frozen CLIP-based representations to identify hateful meme content, while Vid+RM-FT [272] fine-tuned large-scale VLMs (LLaMA-3.2-11B and LLaVA-NextVideo-7B) on HateMM and re-annotated Hateful Memes datasets [169].

Despite this progress, state-of-the-art LLMs and VLMs often require extensive task-specific

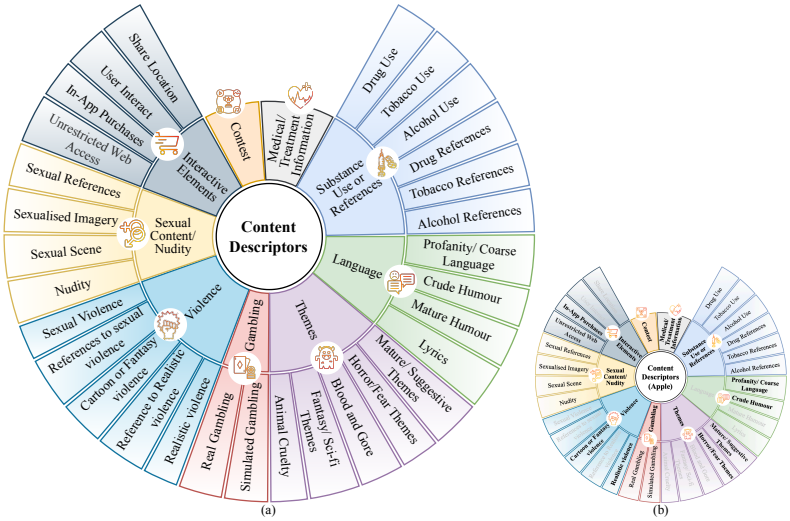


Figure 5.2: (a) Content descriptor taxonomy and (b) mapping to 12 different Apple content rating descriptors

fine-tuning, carefully engineered prompting strategies [270], or additional supervision to generalise reliably to new domains. Moreover, existing work predominantly focuses on social media, memes, or open-domain multimedia, with limited attention to mobile application ecosystems. *To the best of our knowledge, this is the first study to apply VLMs to the detection of undisclosed or misrepresented content rating descriptors in large-scale mobile app marketplaces.*

5.4 Data Pipeline

In this section, we formulate our content rating descriptor (CRD) taxonomy creation and dataset curation for CRD prediction task.

Content descriptor taxonomy: We categorise content descriptors into nine general categories: Language, Themes, Gambling, Violence, Sexual themes and Nudity, Medical treatment Information, Substance and Interactive Elements. Each primary category is further divided into 2-6 sub-categories, resulting in a total of 30 sub-categories, as shown in Figure 5.2.

This taxonomy originates as an amalgamation of Apple content rating guidelines and ACB (which is utilised by Android apps) guidelines, defining all types of content that can be categorised as inappropriate for different age groups. This taxonomy is more concise and generalised, making it more flexible to represent content in both app markets. The taxonomy covers all 12 content descriptors defined in the iOS ecosystem and fully covers all first-layer descriptors specified by the ACB standards. Moreover, the taxonomy can be progressively refined in the following data construction and training process. This adaptability ensures robustness in handling evolving inappropriate content and the mobile app ecosystem.

Curation of training and testing datasets: The manual reviewing process for iOS apps compared to the automated review in Android has resulted in selecting iOS data as ground truth

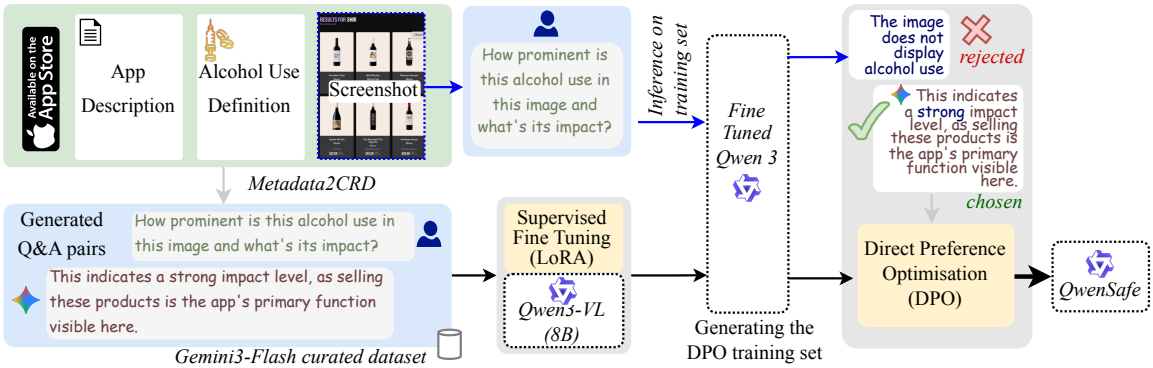


Figure 5.3: The overview of QwenSafe pipeline. The pipeline involves two two-stage process. 1) **metadata2CRD:** constructing QwenSafe training data from Apple app metadata, 2) training Qwen3-VL model.

for verifying Android apps in many prior research [53]. Following this intuition, we construct our training dataset using developer-defined content descriptors, app metadata, and content-rating labels from popular iOS applications. We crawled a large-scale dataset of approximately 1.2 million Apple App Store entries between January and November 2023. From this corpus, we selected the top 780 apps from each content-rating category, excluding the 4+ category because it does not provide descriptor-level information. The selection was stratified across 26 app categories [273], taking the top 30 apps per category, where "top" is determined by sorting apps based on: (i) number of user ratings, (ii) average star rating, and (iii) popularity rank. We posit that highly downloaded applications are less likely to contain incorrect information, as their large user base increases the likelihood of detecting and reporting violations, thereby enhancing the reliability of their content rating labels. For each app, we extract textual information such as short and long app descriptions, visual information such as the app icon and app screenshots, and developer-defined metadata such as content rating label and content rating descriptors.

To evaluate our pipeline, we construct a iOS test set by selecting the next ten most popular apps per category for each content-rating class, yielding a dataset of 981 applications.

5.5 Methodology

In this section, we outline the overall methodology of QwenSafe, which comprises two primary components: (1) training data construction and (2) offline model adaptation for content-rating descriptor (CRD) reasoning. As detailed in Section 5.5.1, we employ Gemini-2.5-Flash to generate descriptor-aligned question–answer pairs for each image, forming a high-quality supervision signal for CRD interpretation. Section 5.5.2 then describes the two-stage training pipeline, where the Qwen3-VL-8B model is adapted through LoRA-based supervised fine-tuning followed by mistake-driven Direct Preference Optimisation to align the model with descriptor-specific rationales.

5.5.1 Construction of Training Data

metadata2CRD: Descriptor-Aligned Q&A dataset Construction: As outlined in Section 5.4, our unified descriptor taxonomy contains 32 fine-grained content descriptors, whereas Apple's taxonomy has only 12 descriptor classes and collapses several categories into single labels. For example, Apple provides only one aggregated descriptor for alcohol, tobacco, or drug use, despite the presence of six separate sub-descriptors in our taxonomy. To reconcile this mismatch, we first align all Apple-provided labels with the corresponding first- and second-layer nodes in our hierarchical taxonomy. This alignment enables a consistent mapping from marketplace-specific descriptor schemes to our unified CRD space. After aligning descriptor categories, we construct a descriptor-definition corpus by compiling formal definitions from ESRB, ACB, and Apple guidelines [41, 38, 274]. Each descriptor is thus associated with a textual definition. For each app, developers provide a set of metadata—most notably, an app description and multiple screenshots—that we treat as multi-modal input features. Rather than relying on traditional VQA-style question–answer pairs [156] that focus solely on observable visual cues, we introduce a structured generation procedure that leverages both mobile app metadata and descriptor-specific definitions to generate question and answer pairs. Given an app screenshot, its description, a target CDR, and a target CRD definition, we prompt the Gemini-2.5-Flash [49] model to analyse the multi-modal input and instruct it to produce chain-of-thought rationales that articulate (i) whether the descriptor is evidenced in the content and (ii) the supporting visual and textual cues. These rationales serve as secondary content-rating descriptor tags that improve interpretability and provide a richer supervision signal. We refer to this procedure as metadata2CRD. Formally, metadata2CRD is composed of four sequential reasoning steps:

- Visual Understanding - The model identifies salient visual cues from the app screenshot that can be leveraged to characterise the content.
- Textual Understanding - The app description is analysed to build an understanding of the app's intent and functionality, complementing the visual evidence.
- Descriptor-Definition Matching - Using the given descriptor definitions, the model evaluates whether the combined visual and textual evidence satisfies the criteria for the target descriptor.
- Q&A Generation - The model summarises the above analyses into structured question–answer pairs that:(a) indicate the presence or absence of the descriptor in the image, and (b) justify this decision by citing relevant cues from both modalities.

This process produces descriptor-aligned conversations that serve as training examples for our supervised fine-tuning and DPO stages. An overview of the workflow can be found in Figure 5.3. From the secondary CRD tags generated by Gemini, we retain only those conversations whose predicted descriptors are consistent with the developer-defined labels. After filtering, our final training corpus comprises 79,500 high-quality Q&A pairs with corresponding images.

5.5.2 Offline Training of QwenSafe

Overview of Qwen3-VL: We adapt Qwen3-VL-8B model [156] to content rating descriptor generation task. Qwen3-VL is an efficient VLM with three components tailored for high-resolution visual understanding and scalable language reasoning. (1) Large Language Model: Qwen3-VL integrates the Qwen3 family of LLMs, providing strong general reasoning capabilities. (2) Vision Encoder: build upon the SigLIP-2 architecture [275] as the visual backbone, to extracts visual cues. (3) MLP-Based Vision–Language Merger: apply a lightweight two-layer MLP to compress visual features into a single visual token aligned with the LLM hidden dimension. In addition, specialized merger modules are introduced to support the DeepStack mechanism [276], enhancing multi-level visual fusion.

Training: Our training pipeline consists of two stages: (1) LoRA-based supervised fine-tuning (SFT) and (2) mistake-driven Direct Preference Optimization (DPO).

We fine-tune the Qwen3-VL-8B-Instruct model using all filtered image–text Q&A pairs generated through our metadata2CRD procedure. Because each training example is produced by jointly reasoning over screenshots, app descriptions, and descriptor definitions, the resulting corpus provides highly structured, descriptor-aligned supervisory signals. During SFT, the model is trained to minimise the standard next-token cross-entropy loss:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta \left(y_t \mid x, y_{<t}\right), \tag{5.1}$$

where T represents the number of output tokens, $y_{<t} = y_1, ..., y_{t-1}$ represents all the output tokens before position t and x represents the input sequence containing both visual tokens and text prompts. p_θ is the probability that the model assigns to the next token y_t given all previous tokens $y_{<t}$, under the current parameters θ. In this phase, the vision encoder and base language model remain fixed, and only the multimodal projection layer and LoRA adapters on selected transformer blocks are updated. This parameter-efficient strategy enables the VLM to learn CRD-specific reasoning, while preserving the pre-trained visual and linguistic representations.

Following the SFT phase, we employ Direct Preference Optimization (DPO) [262] to further align the model with the CRD and enhance predictive accuracy. After completing SFT, we run the fine-tuned model over the entire SFT corpus and sample multiple responses per instance across three inference passes. These samples are compared against the reference answers produced by our metadata2CRD procedure, enabling us to collect both aligned (correct) and misaligned (incorrect) outputs. To identify misaligned responses, we employ Meta-Llama-3-8B-Instruct as a judge. For each item, the judge model is prompted with the original reference answer and the SFT-generated response and asked to assign an alignment score from 0 to 5 (0 indicating complete disagreement and 5 indicating perfect agreement). Responses with scores below 3 are treated as erroneous outputs and selected for DPO training. Each DPO pair thus

consists of: (i) the ground-truth answer from the metadata2CRD process as the winning response, (ii) the misaligned SFT-generated answer as the losing response, and (iii) the associated visual-textual input. The model is then optimized via the standard DPO objective:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\hat{r}_\theta(x, y_w) - \hat{r}_\theta(x, y_l)\right)\right] \tag{5.2}$$

$$\text{where } \hat{r}_\theta(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

Here, π_θ denotes the model being optimized, π_{ref} is the SFT model, and y_w and y_l represent the preferred and dis-preferred responses, respectively. This stage prioritises the model to generate more descriptor-consistent rationales.

5.6 Experimental Setup

Dataset: Using the procedure described in Section 5.5.1, we construct a corpus of 79,500 descriptor-aligned question–answer pairs derived from 2,340 iOS applications. This dataset serves as the training dataset for the SFT of the QwenSafe model. Following SFT, we further obtain 7,356 mistake-focused image–text preference pairs, which are used to train the model through DPO. We used 981 iOS apps as the test dataset to evaluate the QwenSafe model.

Evaluation Metrics: The goal of QwenSafe is to reliably detect the presence of specific content rating descriptors in mobile app metadata. To assess model performance, we formulate the task 1) binary classification for each CRD, where the model's generated textual output is mapped to a positive or negative label and 2) multi-class classification, which predicts the impact as either strong or mild (see Section 5.6.1 for details). For each descriptor, we report precision and recall for both the positive and negative classes. Among these metrics, we prioritise *positive recall* and *negative precision*, as they capture the two most critical behaviours for safety rating: (i) high positive recall ensures that instances containing objectionable or rating-relevant content are correctly identified, while (ii) high negative precision reduces false alarms by ensuring that benign content is classified as harmful. This aligns the evaluation with the practical requirements of content rating systems, which must avoid both missed violations and unnecessary over-flagging.

Baselines: To evaluate the performance of QwenSafe, we compare against several strong vision–language models (VLMs). Specifically, we evaluate: (i) Qwen3-VL-8B [156], the underlying backbone of QwenSafe; (ii) LLaVA-v1.6-Mistral-7B [277], a similarly sized open-source VLM capable of joint image–text reasoning; and (iii) Gemini-2.5-Flash [49], a state-of-the-art multimodal model from Google DeepMind. Gemini-2.5-Flash is included both as a high-capacity reasoning baseline and because it is used in our data construction pipeline for generating SFT training corpora. For each model, we provide the app screenshot and its corresponding textual description as input, and prompt the model to determine the presence or absence of each

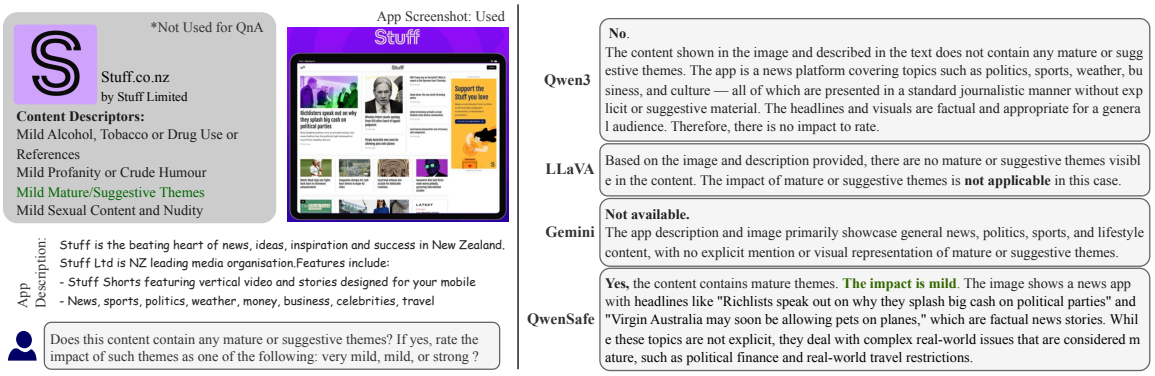


Figure 5.4: Example illustrating model behaviour on the "Mature/Suggestive Themes descriptor". QwenSafe recognises the subtle cues present in both the screenshot and description and accurately labels the impact as mild, demonstrating improved sensitivity to low-intensity content.

content rating descriptor. This setup ensures a consistent and comparable evaluation across all baselines.

5.6.1 Evaluation Results

First, we evaluate binary classification performance across 12 Apple-defined content rating descriptors (**cf.** Figure 5.2 (b)), using app descriptions paired with multiple images (icons and screenshots). A descriptor is considered present if at least one image–text pair is flagged positive. As shown in Table 5.1, QwenSafe substantially improves descriptor detection over baseline VLMs. Relative to Qwen3-VL, our method yields a 111.8% gain in positive class recall, demonstrating that the QwenSafe pipeline reliably identifies descriptor-relevant content while also improving negative class precision by 2.9%, reducing false positives in non-descriptor cases. Compared with LLaVA-v1.6-7B, QwenSafe achieves a 36.1% higher positive recall, underscoring its advantage in identifying safety-critical instances. Further, the lower R^+ and higher P^- indicate that LLaVA and Qwen3 underestimate the descriptor availability in an app. Although LLaVA often identifies descriptor availability, it struggles to assess descriptor severity, indicating that general-purpose VLMs lack specialised alignment for fine-grained content-rating tasks. Gemini-2.5-Flash performs competitively (50.01% R^+; 90.3% P^-), yet QwenSafe surpasses this large-scale model by $\geq 2.1\%$ on both metrics.

Table 5.2 reports multi-class classification performance for mild and strong impact levels across content rating descriptors. Overall, QwenSafe outperforms both Qwen3-VL-8B and Gemini-2.5-Flash in detecting mild and strong descriptor classes, demonstrating improved recall to graded content severity. Qwen3-VL shows notably low performance on the mild class ($R_{mild} = 0.0458$ on average), indicating that it struggles to recognise mild impact descriptors. Although Qwen3 achieves comparatively higher recall for strong indicators ($R_{strong} =$

Content Rating Descriptor	LLaVA		Qwen3		Gemini		QwenSafe	
	R^+	P^-	R^+	P^-	R^+	P^-	R^+	P^-
Alcohol, Tobacco or Drug Use or References	0.1687	0.8552	0.1386	0.8507	0.4337	0.8522	**0.5422**	**0.9115**
Horror or Fear Themes	0.4522	**0.9322**	0.2609	0.9106	**0.4870**	0.9095	0.4000	0.9238
Mature or Suggestive Themes	0.1625	0.7465	0.2438	0.7654	0.5230	0.7589	**0.5548**	**0.7978**
Medical / Treatment Information	0.4505	0.9345	0.3694	0.9255	0.5586	0.9241	**0.5766**	**0.9433**
Profanity or Crude Humour	0.2287	0.8454	0.1011	0.8243	0.4149	0.8254	**0.4628**	**0.8855**
Cartoon or Fantasy Violence	0.3396	0.8867	0.2830	0.8782	**0.5597**	0.8869	0.5220	**0.9110**
Realistic Violence	0.3425	0.9498	0.2603	0.9439	0.5205	0.9478	**0.6027**	**0.9664**
Sexual Content or Nudity	0.3333	0.9016	0.2754	0.8940	0.4275	0.8783	**0.5145**	**0.9229**
Simulated Gambling	**0.4054**	**0.9772**	0.1622	0.9682	0.3243	0.9641	0.3514	0.9744
Real Gambling	0.1579	0.9506	0.1228	0.9487	0.1579	0.9313	**0.2456**	**0.9546**
Unrestricted Web Access	0.7429	0.9701	0.1619	0.9087	**0.8762**	**0.9789**	0.6095	0.9384
Contests	0.7179	**0.9885**	0.5128	0.9802	0.7179	0.9838	**0.7436**	0.9862
Average	0.3752	0.9115	0.2410	0.8999	0.5001	0.9034	**0.5105**	**0.9263**

Table 5.1: Binary classification performance comparison. Positive class recall (R^+) and negative precision (P^-) across content rating descriptors for LLaVA-v1.6-mistral-7b-hf [277], Qwen3-VL-8B [156], Gemini-2.5-Flash [49], and QwenSafe(ours). The binary classification only determines whether a content rating descriptor is available in app metadata or not. Best performance is denoted in bold.

0.4591), QwenSafe still provides a 20.34% absolute improvement in strong-class recall, achieving 0.5525 recall. This behaviour is consistent with the binary results in Table 1, where Qwen3 attains a positive recall of 0.241 and a negative precision of 0.8999, reflecting limited robustness in fine-grained CRD detection. Gemini performs moderately on the mild class and comparatively better on strong indicators. In contrast to QwenSafe, both Gemini and Qwen3 exhibit high precision but comparatively lower recall across both mild and strong indicators, suggesting a limited ability to capture indirectly expressed content descriptors.These patterns reflects the fact that general-purpose multimodal models focus on broad content understanding rather than graded severity differentiation. *While the generic VLM models can often identify the presence of a descriptor, they do not reliably distinguish between its mild and strong variants, as the distinction requires semantic alignment*.

In contrast, QwenSafe achieves the best performance across all CRD categories. It improves average mild-class recall by 62.75% relative to Gemini and achieves the highest overall strong-class recall, exceeding Gemini by 12.64%. One factor affecting mild-class recall is that some apps formally declare CRDs, but their visual and textual metadata do not explicitly depict mild-level cues, making these instances inherently challenging. Despite this, *QwenSafe demonstrates consistently superior discrimination across both severity levels. These gains highlight the effectiveness of our metadata2CRD data construction pipeline and targeted fine-tuning strategy to learn descriptor specific features.*

Content Rating Descriptor	**Qwen3**				**Gemini**				**QwenSafe**			
	P_{mild}	R_{mild}	P_{strong}	R_{strong}	P_{mild}	R_{mild}	P_{strong}	R_{strong}	P_{mild}	R_{mild}	P_{strong}	R_{strong}
Alcohol, Tobacco or Drug Use/ References	0.7143	0.0676	0.6667	0.3333	0.9259	0.3378	0.5000	**0.5000**	0.6952	**0.4932**	0.5294	**0.5000**
Horror or Fear Themes	0.7500	0.0577	0.2273	0.4545	0.7619	0.1538	0.1143	0.3636	0.5000	**0.2308**	0.2857	**0.7273**
Mature or Suggestive Themes	0.8889	0.0348	0.5500	**0.6226**	0.8077	0.2739	0.3571	0.4717	0.3195	**0.3696**	0.3152	0.5472
Medical / Treatment Information	1.0000	0.0337	0.4474	**0.7727**	0.8000	0.1798	0.3810	0.7273	0.2655	**0.3371**	0.3590	0.6364
Profanity or Crude Humour	0.9000	0.0545	0.3333	0.1304	0.8293	0.2061	0.1622	**0.2609**	0.8111	**0.4424**	0.2222	0.0870
Cartoon or Fantasy Violence	0.9048	0.1329	0.2500	0.3750	0.8511	0.2797	0.1667	0.4375	0.5422	**0.3147**	0.2045	**0.5625**
Realistic Violence	0.5000	0.0152	0.0588	0.1429	0.9000	0.1364	0.1071	**0.4286**	0.3559	**0.3182**	0.0517	**0.4286**
Sexual Content or Nudity	1.0000	0.0163	0.3333	**0.8000**	0.9286	0.2114	0.2903	0.6000	0.6324	**0.3496**	0.2500	0.7333
Simulated Gambling	0.0000	0.0000	0.8000	0.5000	0.7500	0.1034	0.6250	0.6250	0.3333	**0.2069**	0.2400	**0.7500**
Average	0.7398	0.0458	0.4074	0.4591	0.8394	0.2091	0.3004	0.4905	0.4950	**0.3403**	0.2731	**0.5525**

Table 5.2: Multi-class classification performance across content rating descriptors. We report mild and strong precision and recall for Qwen3-VL-8B [156], Gemini-2.5-Flash [49], and QwenSafe. We exclude gambling, unrestricted web access, and contests because Apple does not provide impact levels for these descriptors. The best recall for both classes is recorded in bold.

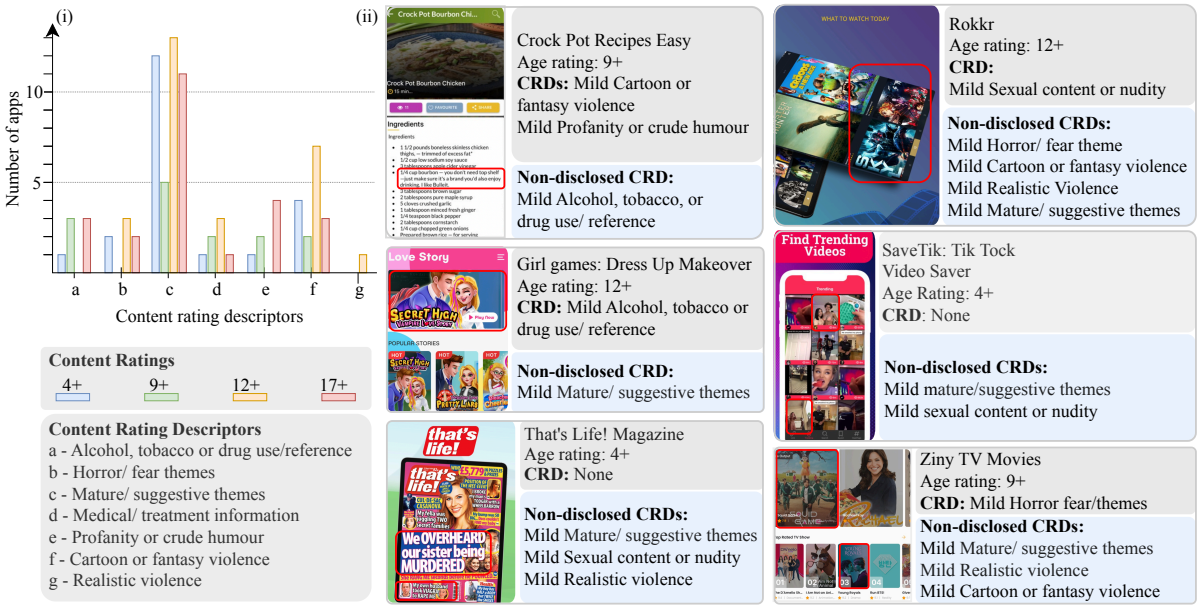


Figure 5.5: Analysis of non-disclosed content rating descriptors (CRDs) identified by QwenSafe. (i) Distribution of applications containing non-disclosed CRDs across Apple age rating categories (4+, 9+, 12+, and 17+) and descriptor types. (ii) Representative examples of applications where QwenSafe detects CRDs that are not declared in the app metadata.

5.6.2 Non-Disclosed Descriptors.

Apple's age-rating system requires developers to disclose the presence of specific content rating descriptors (CRDs). However, some descriptors may appear in the app's visual content but remain unreported in the app metadata. We refer to these as non-disclosed descriptors. To quantify and analyse such cases, we evaluate all apps in our test set using QwenSafe and assess the presence of all 12 Apple-defined CRDs, regardless of what developers reported. Figure 5.5 (i) summarises the number of apps containing non-disclosed CRDs for each descriptor, while Figure 5.5 (ii) presents representative examples identified by QwenSafe. According to Apple's guidelines, apps rated 4+ are expected to contain no objectionable material. However, our analysis reveals 21 apps within the 4+ category that exhibit at least one non-disclosed CRD. Notably, Apple's age rating definitions specify that apps containing mild cartoon or fantasy violence, mild profanity or crude humour, or mild mature, suggestive, or horror-themed content must be rated 9+ or higher. As illustrated in Figure 5.5(ii), we identify multiple 4+ applications that display visible mature or suggestive themes, rendering them inappropriate for the intended age group. We further observe similar inconsistencies in higher age categories. Specifically, within the 9+ category, we identify 14 applications with non-disclosed CRDs. These include applications with alcohol references as well as a media streaming application for which QwenSafe detects additional descriptors, including realistic violence, that are not reflected in the

declared metadata. While Apple permits mild or infrequent mature or suggestive themes for the 9+ category, we identify five applications in which such content is present but not disclosed. Comparable patterns of missing descriptors are also observed in applications rated 12+ and 17+.

Apple states that applications undergo manual review prior to publication on the App Store. Nevertheless, our evaluation indicates that 9.37% of applications in the test set contain at least one non-disclosed CRD. As discussed in Section 5.6.1 and illustrated in Figure 5.4, even state-of-the-art vision–language models such as Gemini and Qwen3 exhibit limited effectiveness in reliably identifying CRDs and their severity levels. This gap introduces potential safety risks, particularly for younger users. *In contrast, QwenSafe demonstrates the ability to systematically detect undisclosed descriptors and align predictions with Apple's content rating definitions, providing a more reliable and policy-aligned solution for this downstream safety-critical task.*

5.7 Conclusion

We introduce QwenSafe, a vision–language model for automatic identification of content rating descriptors in mobile applications, supported by metadata2CRD, a scalable pipeline for constructing descriptor-aligned supervision from app metadata and screenshots. By combining supervised fine-tuning with DPO-based preference alignment, QwenSafe consistently outperforms strong vision–language baselines across 12 Apple-defined descriptors, with particularly large gains in positive-class recall. These results demonstrate the effectiveness of descriptor-aware multimodal alignment for automated content classification and highlight the practical potential of vision–language models for scalable and consistent content rating in large app marketplaces.

Chapter 6

RankOOD - Class Ranking-based Out-of-Distribution Detection

As outlined in the scope of this thesis (**cf.** Section 1.3), this chapter[1] propose an Out-of-Distribution (OOD) detection approach based on training a model with the ListMLE loss, where the model is optimised to preserve the given class-ranking.

6.1 Introduction

Despite their tremendous success, deployment of Deep Neural Networks (DNNs) in critical applications remains restricted due to the out-of-distribution (OOD) problem. For example, a vision classifier installed in an autonomous vehicle may make an inaccurate but overconfident prediction for an object class not seen in the training data. Accurate OOD detection continues to remain a fundamental unresolved problem not only in computer vision, but also in all machine learning [278, 279, 280].

Existing OOD detection approaches can be broadly divided into two categories: post-hoc methods and training-based methods. Post-hoc methods [278, 281, 282, 283, 284] operate on pretrained models and extract OOD signals from their outputs or internal representations, offering simplicity and compatibility with existing networks. In contrast, training-based methods modify the learning process to improve the separability between in-distribution (ID) and OOD data. Among these, outlier-exposure techniques [285, 286, 287, 288] explicitly use auxiliary outlier datasets during training, while training without outliers methods enhance OOD robustness implicitly through regularisation or objective design. Although post-hoc approaches are lightweight and maintain ID performance, training-based approaches typically achieve stronger OOD detection, often at the cost of reduced classification accuracy on ID samples.

[1] This chapter includes the work in: **D. Denipitiyage**, N.Karunanayaka, S. Seneviratne, S. Chawla, "RankOOD - Class Ranking-based Out-of-Distribution Detection", accepted in The IEEE/CVF Conference on Computer Vision and Pattern Recognition.

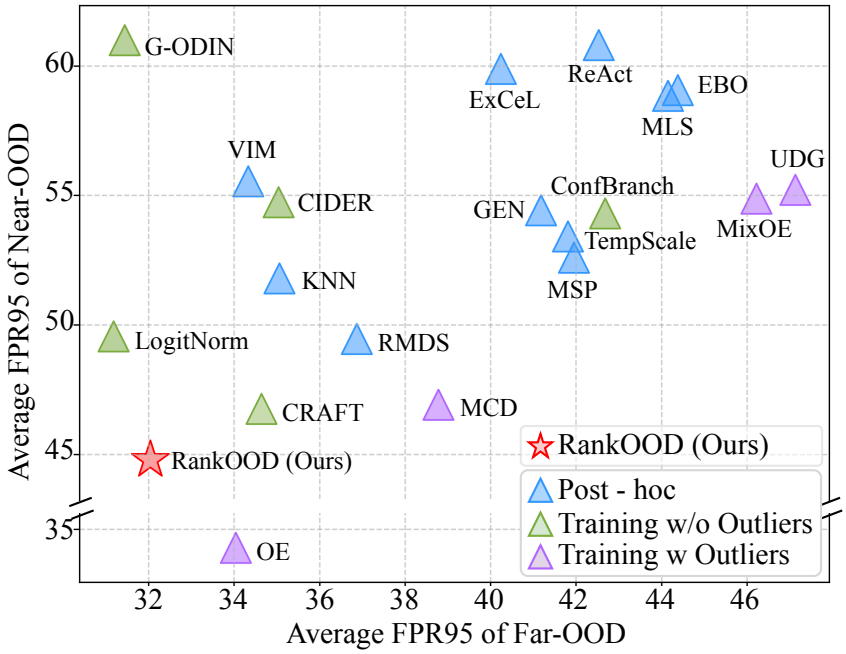


Figure 6.1: .
The performance comparison of average FPR95 on Far-OOD (x-axis) and Near-OOD (y-axis). *RankOOD outperforms all post-hoc and training methods without outliers. Compared to training methods with outliers, its performance is only second to the Outlier Exposure (OE) method.*

Recently, a new line of work has emerged on addressing the OOD problem, which is post-hoc and includes elements of post-training. The key insight is the following:

> A DNN classifier trained for single-class prediction often naturally induces a rank order across classes. An OOD example may be overconfidently assigned to a class label but is unlikely to respect the associated rank order. Thus, if we can train a classifier to strengthen the rank order, then we can achieve both accurate in-distribution prediction and OOD detection.

Several works leverage class ranking patterns as a discriminative signal for OOD detection [289, 290, 291]. For instance, CRAFT [290] fine-tunes pre-trained models to learn class-specific ranking distributions among prediction scores. CRAFT demonstrates that ID samples exhibit more deterministic inter-class ranking relationships, while OOD samples disrupt these patterns. By modelling these relationships via class-dependent probability mass functions (PMFs) and measuring their divergence during inference, CRAFT achieves strong OOD performance. However, this fine-tuning process introduces architectural modifications and ignores the relative ordering of the class ranks.

In this work, we propose RankOOD, a rank-based, listwise OOD detection framework that eliminates the need for fine-tuning or a separate network while preserving the discriminative power of ranking structures. Unlike CRAFT, which models each class ranking as a 2D C×C

PMF matrix (where C is the number of classes), RankOOD directly operates on the raw logits of a pre-trained network and orders them according to predefined rank labels. We employ a Listwise Maximum Likelihood Estimation (ListMLE) [56] objective to learn the rank structure. Unlike pointwise or pairwise objectives, ListMLE treats the entire output list jointly and assigns a probability to each permutation via the Plackett–Luce model [292], maximizing the likelihood of the observed ranking. This allows the model to capture relative ordering dependencies among classes, reflecting their competitive nature in the final decision. Conceptually, our method captures global listwise consistency, modeling how class scores interact within a shared ranking space. This perspective not only simplifies the training pipeline, since no additional fine-tuning or outlier exposure is required, but also provides a canonical ranking structure for OOD detection. We summarize the following contributions,

- We propose RankOOD, a novel method that detects OOD inputs by learning and analysing clase-wise rank order patterns directly from model outputs without fine-tuning or auxiliary outliers.
- We introduce the novel use of a ListMLE objective derived from the Plackett–Luce (PL) model [292], to model entire output rankings for OOD detection, effectively capturing vital inter-class dependencies ignored by prior methods.
- We validate RankOOD by comparing 34 existing methods conducted in the OpenOOD environment [293]. Our method consistently ranks within the top two in near-OOD detection and the top three for far-OOD setting, demonstrating the strong consistency among the evaluated methods. Notably, on the TinyImageNet near-OOD, RankOOD achieves SOTA performance, reducing FPR95 by 4.3% relative to the strongest baseline.

6.2 Related Work

OOD detection methods for image classifiers are primarily grouped into two categories: i) post-hoc inference methods [278, 294, 282] and ii) training-based methods [295, 296, 285].

Post-hoc inference methods: These methods utilise a standard classifier (usually trained with cross-entropy) and compute an OOD score from its outputs or intermediate features. A simple baseline is the Maximum Softmax Probability (MSP) detector, which uses the largest softmax score as the anomaly score [278]. Other classic scores include the Mahalanobis distance [294], which measures deviations from class-conditional feature statistics, and the energy score [282], defined as the negative log-sum-exp of the logits. More recent methods aim to further improve the separability between ID and OOD samples by refining DNN outputs or activations through techniques such as rectification [283], shaping [297], and sparsification [298].

Training-based methods: Following the OpenOOD taxonomy [293], training-based methods can be further grouped into methods without outliers [295, 299, 300] and with outliers [285, 288,

286]. *Training methods without outliers* modify the learning objective using only ID data to improve calibration and robustness. RotPred [296] introduces an auxiliary rotation prediction task, leveraging self-supervision to encourage feature diversity that aids OOD detection. CSI [301] applies contrastive self-supervised learning to cluster ID samples tightly in embedding space. LogitNorm [295] enforces a fixed norm on the output logits during training, preventing overconfident predictions and substantially improving post-hoc OOD separability. These methods offer strong performance without relying on external data, making them practical for closed-set training scenarios.

On the other hand, when auxiliary outlier data are available, models can explicitly learn to assign low confidence to them (i.e., *training methods with outliers*). The outlier exposure method [285] introduces a loss term that penalises overconfident predictions on a diverse auxiliary dataset, encouraging uniform output distributions for OOD samples. Building on OE, MixOE [288] interpolates ID and OOD samples to generate mixed examples that regularise model confidence. Other variants, such as MCD [286] and UDG [287], leverage ensemble disagreement or unsupervised clustering to synthesise pseudo-OOD data.

Each category of methods has its own strengths and trade-offs. Post-hoc approaches are simple to apply and require no retraining, but their effectiveness depends heavily on the quality and calibration of the underlying model. Training-based methods without outliers tend to be more robust and data-efficient, though they may struggle when confronted with unfamiliar, real-world anomalies. Outlier-assisted methods often achieve the strongest results but risk overfitting to the seen outlier data and rely on access to sufficiently diverse and representative outlier examples, which may not always be practical.

Class rank-based OOD detection A recent line of research has leveraged class-rank information to improve OOD detection. For example, ExCeL [291] computes a post-hoc OOD score by combining the maximum logit with a *class rank signature* that captures the probability of each class appearing in subsequent ranks. Extending this idea, CRAFT [290] enhances OOD detection by fine-tuning a pre-trained classifier to sharpen its learned class-rank patterns, which are modelled as PMFs. *Our work, RankOOD, further advances this direction by optimizing class-wise rank structures under the Plackett–Luce formulation [292], using standard vision backbones without architectural changes.*

6.3 Methodology

In this section, we present the overall methodology of RankOOD, which consists of three main steps: i) using a pre-trained model on ID data, we derive a canonical class ranking for each class by solving a rank assignment problem (Section 6.3.1); ii) we then train a new classifier on the ID data using the canonical class rankings as ground truth and the ListMLE loss, referred to as

RankOOD-T, the training process of our framework (Section 6.3.2); and iii) during inference, we compute an OOD score (termed RankOOD-S) based on the predicted ranks and their deviation from the canonical rankings, under the premise that OOD samples deviate more from the canonical class ranking patterns (Section 6.3.3).

6.3.1 Finding Canonical Class Ranks

Using a pre-trained model on in-distribution train data, we first compute a *Rank Probability Matrix* (RPM) [290] for each class $c \in C$, which comprises the probability mass functions (PMFs) across all rank positions, estimated from ID samples that are correctly predicted as class c. Specifically, for each class c, an element $p^c_{i,j}$ within the RPM, $P^c \in \mathbb{R}^{C\times K}$ denotes the probability that class i appears at the $j^{th} \in K$ rank when the input is classified as class c. Thus, each column of the matrix P^c_j represents a PMF over ID classes corresponding to a particular rank position j.

To obtain a consistent canonical class ranking for each top-rank class, we solve an Integer Linear Programming (ILP) problem. That is, we formulate a 0-1 integer linear program that selects one class per rank, ensuring uniqueness and maximizing the total assignment probability. For each class, this yields a consistent ranking that best reflects the trained preference structure.

Given the class c, and its rank probability matrix, $P^c \in \mathbb{R}^{C\times K}$, we introduce binary decision variables,

$$x^c_{i,j} = \begin{cases} 1, & \text{if class } i \text{ is assigned to rank } j, \\ 0, & \text{otherwise.} \end{cases}$$

We compute the consensus ranking by solving the following 0-1 integer linear program:

$$\max_x \sum_{i=1}^{C}\sum_{j=1}^{K} x^c_{i,j} p_{i,j} \tag{6.1}$$

subject to

$$\sum_{i=1}^{C} x^c_{i,j} = 1 \forall j \in [1,K], \quad \sum_{j=1}^{K} x^c_{i,j} \leq 1 \forall i \in [1,C]$$

This yields a valid ranking permutation that maximizes the joint probability of the K ranking structure under the model. Note that the first constraint ensures only one class is selected per rank, and the second constraint ensures that a class is selected at most once, across all considered ranks.

Example: Consider this worked example in a four-class classification problem as shown in Table 6.1. We consider the 100 correctly classified samples of Class 2.

Since we are considering only 100 correctly classifying samples of Class 2, the frequency

f^2_{ij} for (2,0), i.e., the number of samples where Class 2 is in the 0^{th} rank is 100. This also means, there are no samples with any other class in rank 0, i.e., $f^2_{i0}\ \forall\ i\backslash\{2\}$ is 0. Also means that Class 2 can not appear in any other position than rank-0, i.e., $f^2_{2j}\ \forall\ j\backslash\{0\}$ is 0.

Consider the other f^2_{ij} values as filled in the table. Note that the j^{th} column sum, i.e., $\sum_{j=1}^{4} f^2_{ij}$ is 100. Next, the relative frequency can be converted to probabilities, p^2_{ij}, as shown inside brackets in the table. The shaded cells of this table, ignoring the trivial row and column for class 2, form the RPM $\in 4 \times 4$ for Class 2.

Note that in this example, for Class 2, rank 1 is dominated by Class 1, and rank 2 is dominated by Class 3, and rank 3 is dominated by Class 4. Though the canonical class is clear cut in this example, in practice, as the number of classes increases, RPMs get noisier, leading to ties between classes in some ranks. Our ILP solution ensures that the best representative ranking order is selected for each class.

		Rank (j)			
		0	1	2	3
Class (i)	1	0	80 (0.80)	15 (0.15)	5 (0.05)
	2	100	0	0	0
	3	0	10 (0.10)	75 (0.75)	15 (0.15)
	4	0	10 (0.10)	10 (0.10)	80 (0.80)

Table 6.1: An example RPM for Class 2, in a four class classification problem

6.3.2 Ordered Preference Learning

Next, leveraging the ranking information generated through ILP, we train a model using the hybrid objective combining cross-entropy (CE) loss and Listwise Maximum Likelihood Estimation (ListMLE) [56]. While CE encourages the model to predict the correct top class, ListMLE enforces consistency across the entire ordered label sequence by maximizing the likelihood of the observed ranking under the Plackett–Luce formulation [292]. The ListMLE loss is defined as:

$$\mathcal{L}_{\text{ListMLE}} = -\sum_{i=0}^{K-1}(l_{\pi_i} - log(\sum_{j=i}^{K-1}\exp(l_{\pi_j}))) \tag{6.2}$$

equivalently expressed under the Plackett–Luce model as

$$\mathcal{P}(\pi|l) = \prod_{i=0}^{K-1}\frac{\exp(l_{\pi_i})}{\sum_{j=i}^{K-1}\exp(l_{\pi_j})}, \quad \mathcal{L}_{\text{ListMLE}} = -log[\mathcal{P}(\pi|l)] \tag{6.3}$$

Here, $\pi = (\pi_0, \pi_1, \ldots, \pi_{K-1})$ denote the ground-truth ranking where π_i is the class index assigned to rank i and l_{π_i} denote the logits of the class at rank i produced by the model. The Eq. 6.3 defines a parameterized exponential probability distribution over all the permutations

given the predicted result by the model $\mathcal{F}_\theta$, and defines the loss function (Eq. 6.2) as the negative log likelihood of the ground truth ranks. This formulation enforces relative ordering ($l_{\pi_0} > \ldots > l_{\pi_i} > \ldots > l_{\pi_{K-1}}$) consistency across the entire ranked list rather than optimizing only the top-1 classification decision. Unlike conventional CE loss, which encourages only the correct label to have maximal score, ListMLE optimizes the full permutation likelihood, ensuring a coherent and structured logit hierarchy. This property makes ListMLE particularly suitable when the model must preserve rich class-relation structure and when OOD detection depends on stable, interpretable score ordering beyond the top prediction. We empirically show that it is not necessary to train the model on the entire rank sequence when only a subset of ranks is sufficient for the downstream OOD score and we selected top-k ranks and lowest-k ranks. However, training on a subset of ranks alone does not constrain the absolute ordering of unsupervised middle positions, nor does it guarantee that the l_{π_0} is globally maximal among all C logits. Therefore, to ensure that the predicted top class is indeed the argmax of the full logit vector, we therefore complement the ListMLE loss with a cross-entropy term,

$$\mathcal{L}_{RankOOD-T} = \mathcal{L}_{CE} + \alpha \mathcal{L}_{\text{ListMLE}} \tag{6.4}$$

The hyperparameter α balances the trade-off between the two objectives.

6.3.3 Detecting OOD Samples

To effectively distinguish ID samples from OOD inputs, we leverage the structure of rank-based logits learned by our RankOOD-T model. First, we derive class-dependent logit threshold profiles from confident training samples, capturing characteristic logit decay across ranks for correctly classified samples. Second, at test time, we compute a RankOOD-S that penalizes deviations from both the expected ranking order and the reference logit thresholds.

Logit threshold Profile - Ref: We construct a reference threshold profile for each class based on the logit values of correctly predicted training samples at rank-0. Among these, we further retain only those samples that satisfy the condition that at least N rank positions yield correct predictions, where N is chosen such that each class has at least one qualifying training sample. For each rank position i, we compute the class-specific reference logit threshold, Ref_i^c as the empirical 95^{th}-percentile of logits. In Section 6.5.4 we explain the percentile selection using the validation set provided by the OpenOOD benchmark [293].

RankOOD-S: Let x denote the logit vector for a given test input, we first determine its predicted class label $\hat{c}$ (i.e., the class related to the highest logit value in x) and obtain the expected ranking order $\pi^{\hat{c}}$ for the predicted class $\hat{c}$. Let $\bar{\pi}$ denote the ranking predicted by the input sample. For each rank position i, we introduce a penalty term if the model misorders the rank, $\bar{\pi}_i \neq \pi_i^{\hat{c}}$. Since the Plackett–Luce objective couples rank i with all preceding ranks $[0, i]$, an

incorrect rank at position i contributes to a deviation in logits in previous ranks. Therefore, we define a cumulative margin penalty on logits in each expected rank i:

$$\delta_{\pi_i^{\hat{c}}} = \gamma^r, \quad r = \sum_{j=i}^{K-1} 1[\pi_j^{\hat{c}} \neq \bar{\pi}_j] \tag{6.5}$$

where $\gamma \geq 1$. Finally, the RankOOD-S measures the weighted deviation of penalized logits from the class-specific reference threshold profile:

$$\text{RankOOD-S} = \sum_{i=0}^{K-1} w_i \log\left(\text{softmax}(\mathbf{u})\right)_i \tag{6.6}$$
$$\text{where} \quad u_i = \frac{x_{\pi_i^{\hat{c}}}}{\delta_{\pi_i^{\hat{c}}}} - Ref_i^{\hat{c}} \quad \forall i \in \{0, \ldots, K-1\}$$

where w_i is learned via linear regression on validation data to maximize separation between ID and OOD samples. Intuitively, ID samples are expected to exhibit a high rank-0 logit value followed by a monotonically decreasing logit sequence that maintains the logit ordering structure across ranks. Since ListMLE in RankOOD-T loss enforces a sequential scoring dependency, a single mid-rank violation reveals inconsistency in the entire confidence trajectory, making the method inherently sensitive to OOD distortions. An example of RankOOD-S is provided in the Appendix C.5.

6.4 Experiments

Datasets: We use CIFAR-10/100 [302] and ImageNet-200 (a.k.a., TinyImageNet) [303] as in-distribution (ID) datasets for our experiments. Following the OpenOOD benchmark [293], we evaluate each ID dataset against both near-OOD and far-OOD test datasets. Specifically, for CIFAR-10, the near-OOD datasets include CIFAR-100 and TinyImageNet, while for CIFAR-100, CIFAR-10 and TinyImageNet are used as near-OOD. For both CIFAR-10 and CIFAR-100, the far-OOD group comprises MNIST [304], SVHN [305], Textures [306], and Places365 [307]. For ImageNet-200, SSB-hard [308] and NINCO [309] serve as near-OOD datasets, whereas iNaturalist [310], Textures [306], and OpenImage-O [284] form the far-OOD group. To ensure consistency, we adopt the same training, validation, and testing splits provided by the OpenOOD benchmark.

Evaluation Metrics: Similar to previous work, we use two standard metrics to evaluate OOD detection performance: (1) FPR95, the false positive rate for OOD samples when the true positive rate for ID samples is 95%; and (2) AUROC, the area under the receiver operating characteristic curve, which measures the detector's ability to distinguish between ID and OOD samples across all thresholds. We report the mean and standard deviation of these metrics over three

independent runs with different random seeds.

Baselines: We compare RankOOD against a comprehensive set of 34 OOD detection methods. We categorise the baselines into three major categories based on the inference methods and the training methods: post-hoc methods, training methods without outliers, and training methods with outliers. Among post-hoc methods, we include classical approaches such as MSP [278] and Energy-based OOD detection (EBO) [282], as well as more recent techniques like ReAct [283], ASH [297], and GEN [311]. Training-based methods without auxiliary outlier data include ConfBranch [312], G-ODIN [299], and LogitNorm [295]. Conversely, methods that utilise auxiliary outlier data for training include Outlier Exposure (OE) [285], MCD [286], UDG [287], and MixOE [288]. Finally, we compare our method against two existing rank-based methods, ExCeL [291] and CRAFT [290].

Implementation Details: For all experiments, we use pre-trained ResNet-18 models [57] as the backbone. For both CIFAR datasets, each model is trained for 500 epochs, while the Tiny-ImageNet model is trained for 300 epochs using RankOOD-T loss. All models use the SGD optimizer with a momentum of 0.9, an initial learning rate of 0.1, and a cosine annealing decay schedule. In our method, we perform hyperparameter sweeps over the loss weighting parameter α in Equation 6.4 as well as over the reference logit threshold, Ref. We set α to $0.8, 1.0$, and 0.5 for CIFAR-10, CIFAR-100, and TinyImageNet, respectively, and select the 95^{th} percentile of the logit distribution as the reference logit threshold for all datasets. The CIFAR-10 network is trained using all 10 ranks, whereas the CIFAR-100 and TinyImageNet models are trained using the top 10 and bottom 10 ranks generated through the ILP procedure. Hyperparameter search details are provided in Section 6.5.4. Additional details on GPU hours, ILP complexity, and ILP runtime are provided in the Appendix C.4.

6.5 Results

We summarize near-OOD results in Table 6.2 and far-OOD results in Table 6.3 on CIFAR-10/100 and TinyImageNet. For brevity, we present average near- and far-OOD performance across all benchmarks for each ID dataset (Section 6.5.1), as well as the overall OOD detection performance. Complete results for all 34 methods, including per-dataset performance, are provided in the Appendix C.1 and Appendix C.2.

6.5.1 Comparison with SOTA Methods

Across the average performance over all benchmarks, RankOOD consistently ranks among the top three methods, achieving the second-best near-OOD (**cf.** Table 6.2) in terms of both AUROC and FPR95. RankOOD is also the third-best far-OOD (**cf.** Table 6.3) performance. Similarly,

Table 6.2: Performance comparison in *near-OOD detection*. For each column, the top five methods are marked in **bold**.

Method	CIFAR-10		CIFAR-100		ImageNet-200		Average	
	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓	AUROC↑	FPR95↓
Post-hoc inference methods								
MSP [278]	88.03 ± 0.25	48.17 ± 3.92	80.27 ± 0.11	**54.80 ± 0.33**	83.34 ± 0.06	54.82 ± 0.35	83.88	52.60
TempScale [281]	88.09 ± 0.31	50.96 ± 4.32	80.90 ± 0.07	**54.49 ± 0.48**	**83.69 ± 0.04**	54.82 ± 0.23	84.23	53.42
RMDS [313]	89.80 ± 0.28	38.89 ± 2.39	80.15 ± 0.11	55.46 ± 0.41	82.57 ± 0.25	**54.02 ± 0.58**	84.17	**49.46**
EBO [282]	87.58 ± 0.46	61.34 ± 4.63	**80.91 ± 0.08**	55.62 ± 0.61	82.50 ± 0.05	60.24 ± 0.57	83.66	59.07
ReAct [283]	87.11 ± 0.61	63.56 ± 7.33	80.77 ± 0.05	56.39 ± 0.34	81.87 ± 0.98	62.49 ± 2.19	83.25	60.81
MLS [314]	87.52 ± 0.47	61.32 ± 4.62	**81.05 ± 0.07**	55.47 ± 0.66	82.90 ± 0.04	59.76 ± 0.59	83.82	58.85
VIM [284]	88.68 ± 0.28	44.84 ± 2.31	74.98 ± 0.13	62.63 ± 0.27	78.68 ± 0.24	59.19 ± 0.71	80.78	55.55
KNN [315]	90.64 ± 0.20	34.01 ± 0.38	80.18 ± 0.15	61.22 ± 0.14	81.57 ± 0.17	60.18 ± 0.52	84.13	51.80
ASH [297]	75.27 ± 1.04	86.78 ± 1.82	78.20 ± 0.15	65.71 ± 0.24	82.38 ± 0.19	64.89 ± 0.90	78.62	72.46
GEN [311]	88.20 ± 0.30	53.67 ± 3.14	**81.31 ± 0.08**	**54.42 ± 0.33**	**83.68 ± 0.06**	55.20 ± 0.20	**84.40**	54.43
ExCeL [291]	86.89 ± 0.23	66.55 ± 0.43	80.70 ± 0.06	55.21 ± 0.56	82.40 ± 0.04	57.90 ± 0.40	83.33	59.89
Training methods without outliers								
RankOOD (ours)	90.21 ± 0.41	**31.72 ± 0.67**	80.67 ± 0.40	**52.59 ± 0.75**	**85.30 ± 0.18**	**50.05 ± 0.16**	**85.39**	**44.79**
CRAFT [290]	**91.11 ± 0.04**	**31.94 ± 1.41**	80.90 ± 0.33	53.73 ± 0.62	**83.65 ± 0.41**	**54.62 ± 0.57**	**85.22**	46.76
ConfBranch [312]	89.84 ± 0.24	31.28 ± 0.66	71.60 ± 0.62	70.21 ± 0.83	79.10 ± 0.24	61.44 ± 0.34	80.18	54.31
G-ODIN [299]	89.12 ± 0.57	45.54 ± 2.52	77.15 ± 0.28	67.58 ± 0.98	77.28 ± 0.10	69.87 ± 0.46	81.18	61.00
LogitNorm [295]	**92.33 ± 0.08**	**29.34 ± 0.81**	78.47 ± 0.31	62.89 ± 0.57	82.66 ± 0.15	56.46 ± 0.37	**84.49**	**49.56**
CIDER [300]	**90.71 ± 0.16**	32.11 ± 0.94	73.10 ± 0.39	72.02 ± 0.31	80.58 ± 1.75	60.10 ± 0.73	81.46	54.74
Training methods with outliers								
OE [285]	**94.82 ± 0.21**	**19.84 ± 0.95**	**88.30 ± 0.10**	**30.73 ± 0.11**	**84.84 ± 0.16**	**52.30 ± 0.67**	**89.32**	**34.29**
MCD [286]	**91.03 ± 0.12**	**30.17 ± 0.06**	77.07 ± 0.32	55.88 ± 0.85	83.62 ± 0.09	**54.71 ± 0.83**	83.91	**46.92**
UDG [287]	89.91 ± 0.25	35.34 ± 0.95	78.02 ± 0.10	61.42 ± 0.48	74.30 ± 1.63	68.89 ± 1.72	80.74	55.22
MixOE [288]	88.73 ± 0.82	51.45 ± 7.78	**80.95 ± 0.20**	55.22 ± 0.49	82.62 ± 0.03	57.97 ± 0.40	84.10	54.88

RankOOD outperforms all prior rank-based approaches, CRAFT [290] and ExCel [291] in both near- and far-OOD settings, achieving an average FPR95 reduction of 7.51% for far-OOD and 4.21% for near-OOD, while outperforming both CIFAR-100 and TinyImageNet datasets and lies on-par with CIFAR-10 performance.

In the near-OOD setting (**cf.** Table 6.2), the only method surpassing RankOOD is OE [285], which benefits from access to outlier samples during training—an assumption RankOOD does not make. Notably, in the near-OOD setting, RankOOD achieves SOTA performance on TinyImageNet, improving AUROC by 0.50% and reducing FPR95 by 4.3% compared to the strongest baseline. This highlights the value of incorporating semantic rank structure when detecting OOD samples in high-cardinality label spaces. Furthermore, among methods that do not utilise outliers, RankOOD obtains the best FPR95 on CIFAR-100, outperforming GEN [311] by 3.36%.

As shown in Table 6.3, despite being on par with top five methods for CIFAR-10, RankOOD ranks second on CIFAR-100 with 83.63% AUROC and 47.44% FPR95, trailing only G-ODIN [299]. Nonetheless, RankOOD outperforms G-ODIN by 8.12% in TinyImageNet FPR95, and G-ODIN performs worst in near-OOD setting for all the datasets. Moreover, in far-OOD setting, RankOOD outperforms all outlier-exposure-based methods on CIFAR-100 and TinyImageNet across all metrics.

6.5.2 Qualitative Study of Rank Distributions

To analyse how rank-based training influences logit distributions, we examine the class-conditional logit distributions across different rank positions on CIFAR-100 under (a) standard cross-entropy

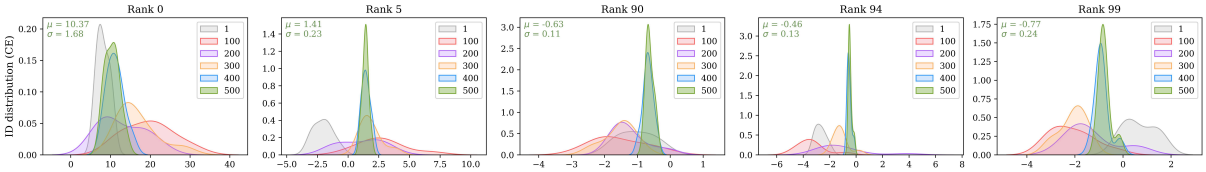
Table 6.3: Performance comparison in *far-OOD detection*. For each column, the top five methods are marked in **bold**.

Method	CIFAR-10		CIFAR-100		ImageNet-200		Average	
	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓
Post-hoc inference methods								
MSP [278]	90.73 ± 0.43	31.72 ± 1.84	77.76 ± 0.44	58.70 ± 1.06	90.13 ± 0.09	35.43 ± 0.38	86.21	41.95
TempScale [281]	90.97 ± 0.52	33.48 ± 2.39	78.74 ± 0.51	57.94 ± 1.14	90.82 ± 0.09	34.00 ± 0.37	86.84	41.81
RMDS [313]	92.20 ± 0.21	25.35 ± 0.73	**82.92 ± 0.42**	52.81 ± 0.63	88.06 ± 0.34	32.45 ± 0.79	87.73	36.87
EBO [282]	91.21 ± 0.92	41.69 ± 5.32	79.77 ± 0.61	56.59 ± 1.38	90.86 ± 0.21	34.86 ± 1.30	87.28	44.38
ReAct [283]	90.42 ± 1.41	44.90 ± 8.37	80.39 ± 0.49	54.20 ± 1.56	**92.31 ± 0.56**	28.50 ± 0.95	87.71	42.53
MLS [314]	91.10 ± 0.89	41.68 ± 5.27	79.67 ± 0.57	56.73 ± 1.33	91.11 ± 0.19	34.03 ± 1.21	87.29	44.15
VIM [284]	93.48 ± 0.24	25.05 ± 0.52	81.70 ± 0.62	**50.74 ± 1.00**	91.26 ± 0.19	**27.20 ± 0.30**	88.81	**34.33**
KNN [315]	92.96 ± 0.14	24.27 ± 0.40	**82.40 ± 0.17**	53.65 ± 0.28	**93.16 ± 0.22**	**27.27 ± 0.75**	**89.51**	35.06
ASH [297]	78.49 ± 2.58	79.03 ± 4.22	80.58 ± 0.66	59.20 ± 2.46	**93.90 ± 0.27**	**27.29 ± 1.12**	84.32	55.17
GEN [311]	91.35 ± 0.69	34.73 ± 1.58	79.68 ± 0.75	56.71 ± 1.59	91.36 ± 0.10	32.10 ± 0.59	87.46	41.18
ExCeL [291]	91.69 ± 0.18	40.03 ± 0.84	**82.04 ± 0.90**	**52.24 ± 1.90**	91.97 ± 0.27	28.45 ± 0.80	88.57	40.24
Training methods without outliers								
RankOOD (ours)	93.19 ± 0.84	20.96 ± 2.55	**83.63 ± 1.06**	**47.44 ± 0.80**	92.14 ± 0.20	**27.73 ± 0.33**	**89.65**	**32.04**
CRAFT [290]	93.94 ± 0.20	**19.40 ± 0.88**	82.03 ± 0.34	**51.86 ± 0.49**	90.88 ± 0.89	32.67 ± 1.13	**88.95**	34.64
ConfBranch [312]	92.85 ± 0.29	21.48 ± 0.94	68.90 ± 1.83	71.82 ± 3.39	90.43 ± 0.18	34.75 ± 0.63	84.06	42.68
G-ODIN [299]	**95.51 ± 0.31**	21.45 ± 1.91	**85.67 ± 1.58**	**42.68 ± 3.19**	**92.33 ± 0.11**	30.18 ± 0.49	**91.17**	**31.44**
LogitNorm [295]	**96.74 ± 0.06**	**13.81 ± 0.20**	81.53 ± 1.26	53.61 ± 3.45	**93.04 ± 0.21**	**26.11 ± 0.52**	**90.44**	**31.18**
CIDER [300]	**94.71 ± 0.36**	**20.72 ± 0.85**	80.49 ± 0.68	54.22 ± 1.24	90.66 ± 1.68	30.17 ± 2.75	88.62	35.04
Training methods with outliers								
OE [285]	**96.00 ± 0.13**	**13.13 ± 0.53**	81.41 ± 1.49	54.82 ± 2.79	89.02 ± 0.18	34.17 ± 0.56	88.81	**34.04**
MCD [286]	91.00 ± 1.10	32.03 ± 4.21	74.72 ± 0.78	54.39 ± 1.34	88.94 ± 0.10	29.93 ± 0.30	84.89	38.78
UDG [287]	**94.06 ± 0.90**	**20.35 ± 2.41**	79.59 ± 1.77	59.00 ± 3.35	82.09 ± 2.78	62.04 ± 5.99	85.25	47.13
MixOE [288]	91.93 ± 0.69	33.84 ± 4.77	76.40 ± 1.44	63.88 ± 2.48	88.27 ± 0.41	40.93 ± 0.29	85.53	46.22

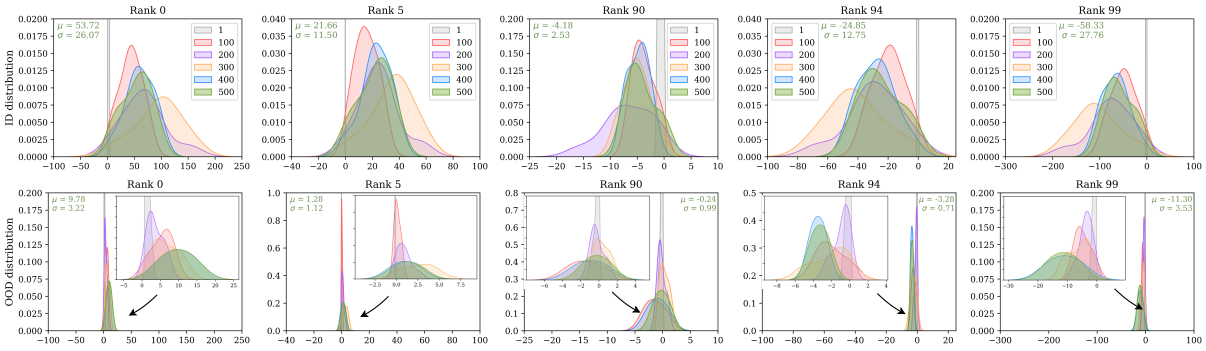
(CE) and (b) RankOOD training. For a given predicted class, we sample logits corresponding to five rank positions (rank-0, three intermediate ranks, and rank-99) and measure their evolution across training epochs. Next, we compare these distributions for in-distribution (ID) and out-of-distribution (OOD) samples.

As shown in Figure 6.2(b), after convergence, the mean logits at ranks 0, 90, and 99 are approximately 53.72, -4.18, and -58.33, respectively, yielding a consistent minimal average rank separation margin of roughly 25 (difference of mean values between distributions). This is because ListMLE optimises the pairwise logit ordering likelihood and therefore depends only on the relative margin between logits of adjacent ranks, penalising margin violations more strongly than large positive margins [56]. Consequently, the optimisation explicitly prioritises preserving the correct ordering rather than pushing the correct class logit arbitrarily high. ListMLE progressively induces a structured semantic hierarchy along the rank dimension: logits associated with semantically closer classes remain comparatively high, while less similar classes are increasingly suppressed, leading to a stable and widening margin across the ranking spectrum. In contrast, low ranks (semantically nearest neighbors) in the CE model initially remain relatively higher, their separation decreases as training progresses, leading to substantial overlap in the logit distributions for mid- and high-rank classes and loses semantic ordering information. Moreover, as shown in Figure 6.2(a), CE induces rank distribution compression, where semantically dissimilar classes become difficult to distinguish in later rank positions.

In the bottom row of Figure 6.2(b), we report the rank-wise logit distributions for OOD samples. Unlike ID samples, OOD logits cluster near zero with substantially lower variance,



(a) CIFAR100 trained using standard cross-entropy CE loss - Logit distribution for different ranks



(b) CIFAR100 trained using ListMLE by progressive enforcement of ranking - logit distribution. (top): In distribution samples (bottom): Out of distribution samples.

Figure 6.2: Logit distributions at selected rank positions for predicted class 82 on CIFAR100. *Across training epochs, (b) ID samples show increasing separation across rank positions, while (b) OOD samples remain concentrated near zero with minimal separation.* OOD samples are shown for comparison only.

forming a compact distribution. Nonetheless, a small yet consistent average margin between ranks remains. This is because ListMLE enforces relative logit margins only for ID samples, which indirectly shapes the geometry of the classifier weights. When an OOD feature lies closer to certain ID class regions, the corresponding logits reflect this proximity, producing slight rank-dependent variation. However, as shown in the Section 6.5.3 and Figure 6.3, OOD samples do not maintain the expected class ranking order for the predicted class.

Overall, the logit magnitude between the ID and OOD in different ranks enables effective OOD detection: ID samples maintain higher, stable rank-dependent margins, while OOD samples remain near zero across ranks.

6.5.3 Logit Ordering Consistency Explains Improved OOD Detection under RankOOD

The left column of Figure 6.3 compares the distributions of MSP and RankOOD-S for CIFAR-10 (ID) and SVHN (OOD) samples under models trained with CE and RankOOD-T losses. Under CE training, MSP (**cf.** Figure 6.3(a)) achieves an FPR95 of 24.24 on SVHN, while RankOOD reduces it by 37.12% (**cf.** Figure 6.3(c)). The RankOOD-T loss explicitly enforces a consistent ranking structure among logits through ListMLE supervision. This encourages the logits to maintain smooth, monotonic decay rather than allowing a single confidence spike. Consequently, even when an OOD input produces a high top-1 logit, its intermediate logits tend to deviate from the canonical decay pattern observed in ID data. This structural deviation

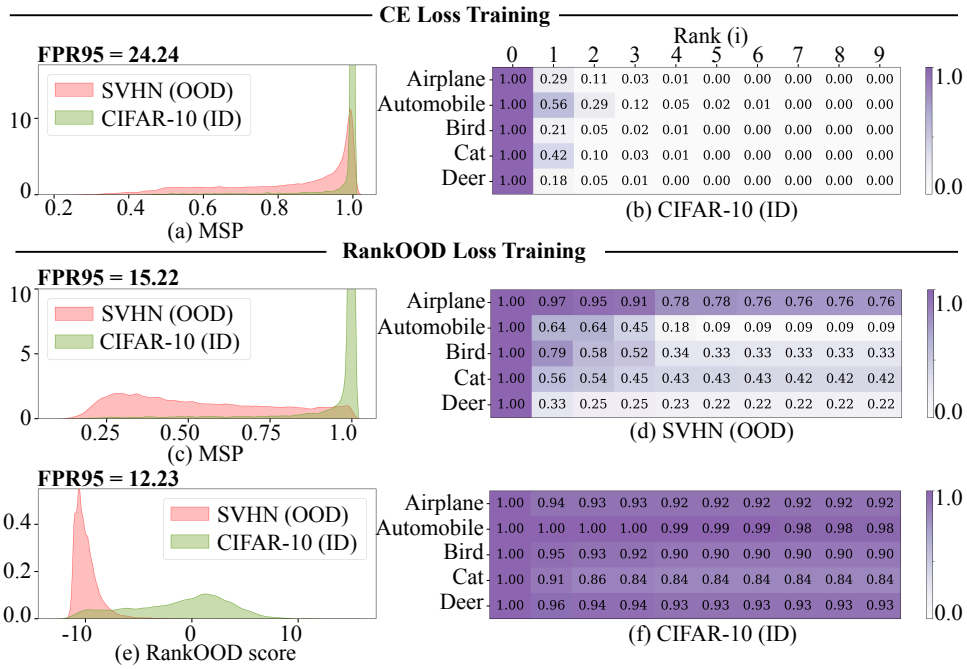


Figure 6.3: Left: Distributions of RankOOD and MSP scores for CIFAR-10 (ID) and SVHN (OOD) samples under models trained with CE and RankOOD losses. Right: Conditional probability matrix (CP) of rank position i given that all prior ranks have been correctly predicted.

reduces the overall MSP score, shifting OOD MSP scores away from 1 and reducing FPR95 for MSP under RankOOD-T. Moreover, the RankOOD-S further decreases FPR95 to 12.23 (Figure 6.3(e)). This occurs because the RankOOD-S amplifies this effect by penalizing rank-order violations and deviations from class-specific logit thresholds.

To analyse how RankOOD leverages ranking consistency, the right column shows conditional probability (CP) matrices capturing $P(rank\ i \mid ranks < i\ are\ correct)$. We report three cases: (b) CIFAR-10 (ID) under CE training, (d) SVHN (OOD) under RankOOD-T, and (f) CIFAR-10 (ID) under RankOOD-T. Due to space constraints, we report CP matrices for five classes, and provide the complete figures in the Appendix C.6. Each CP matrix illustrates the model's ability to preserve the class-wise canonical rank order across ranks. While the CE-trained model (**cf.** Figure 6.3(b)) shows weak rank-consistency, RankOOD-T induces sequential dependencies between ranks, yielding high conditional probabilities (> 0.84) for ID data (**cf.** Figure 6.3(f)). In contrast, OOD samples under RankOOD-T loss exhibit early rank-order violations, accumulating higher penalties δ_{x_i}, in Equation 6.5 and causing higher deviation from the reference logit thresholds Ref_i. These deviations contribute to RankOOD's superior sensitivity to OOD distortions.

6.5.4 Ablation Study

Subset of Ranks: In Figure 6.4 (left), we compare the OOD detection performance on CIFAR-

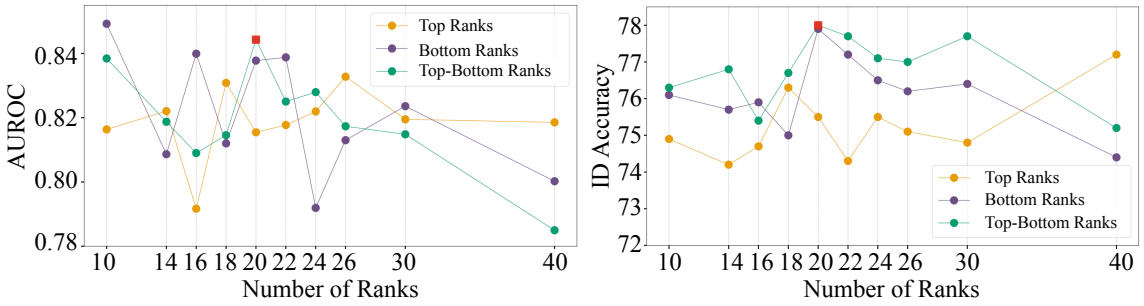


Figure 6.4: OOD and ID detection performance on CIFAR-100 when training with different rank subsets. (left): AUROC (right):ID accuracy. Top Ranks uses the top$-N$ class ranks, Bottom Ranks uses the lowest $N-1$ ranks along with rank-0, and Top-Bottom samples half of the ranks from the top and half from the lowest ranks.

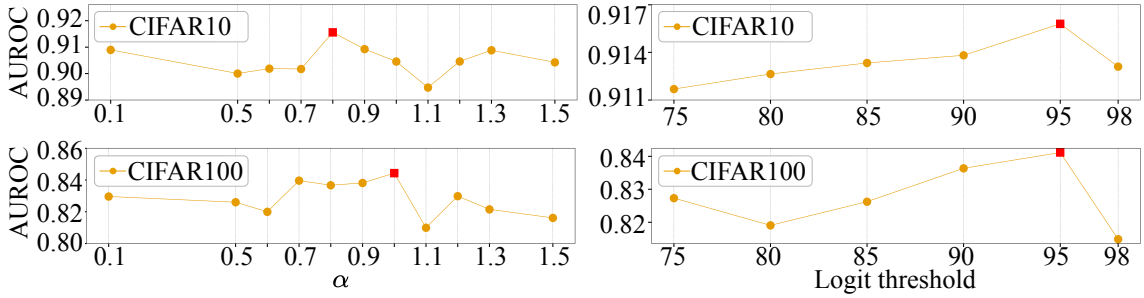


Figure 6.5: OOD detection performance on CIFAR-100 with respect to α and logit threshold (Ref) hyper-parameters.

100 when training with three types of rank subsets; Top ranks subset uses the top-N class ranks, Bottom ranks subset uses the lowest N-1 ranks with rank-0 and Top-Bottom subset consists of the first half and second half of the previous two subsets, respectively. Since the selection of rankings is not only crucial for OOD detection performance but also for ID accuracy, we illustrate the right sub-figure according to the same three subsets. As shown, using only the Top ranks yields lower ID accuracy since it focuses on separating semantically similar classes. In contrast, including Bottom or both Top–Bottom ranks improves ID accuracy by exposing the model to semantically dissimilar classes. Both Top and Bottom rank subsets exhibit unstable AUROC when trained with a small number of ranks. This behaviour arises because the Top ranks primarily distinguish semantically similar class variations, leading to dominance of near-OOD samples, whereas the Bottom ranks, in high-level learns to differentiate highly dissimilar classes compared to rank-0 via logit separations, reducing sensitivity to near-OOD regions. When selecting Top-Bottom ranks, after a certain point, the performance decreases as the ranks introduce more noise. Empirically, selecting the top 10 and bottom 10 ranks (N=20) achieves the best balance - maximising ID accuracy and maintaining high AUROC. This demonstrates that enforcing Listwise ranking across semantically diverse ranks enhances OOD discrimination. We observed these numbers based on an $\alpha = 1.0$ and a logit threshold of 0.95.

Ablation of α and Logit threshold: In Figure 6.5, we present the ablation results of selecting the alpha in Equation 6.4 and the logit threshold (Ref) in the RankOOD-S. We conducted

AUROC tests with different alpha and logit thresholds. For both datasets, $\alpha = 0.1$ shows high AUROC as the loss function is closer to CE loss, and the highest AUROC is achieved at 0.8 for CIFAR-10 and 1.0 for CIFAR-100. With both datasets, we observe AUROC increasing with the increments in the logit threshold, peaking at around 0.95 and starting to decline as the increasing Ref increases the gap between OOD rank logit and threshold in lower ranks, whereas ID samples exhibit a lower gap and vice versa for higher ranks. Consequently, the AUROC decreases once Ref surpasses this threshold, as the threshold becomes the maximum logit value of the given rank. This variation demonstrates significant performance improvement.

Effectiveness of RankOOD-T: We evaluate RankOOD-T with various logit-based OOD scores and consistently outperform standard CE training. These results suggest that primary gains are due to the RankOOD-T objective ; detailed results are provided in Appendix C.3.

6.6 Concluding Remarks

In this chapter, we presented a novel OOD detection framework, *RankOOD*, which leverages class-wise logit ranking structures to enhance the detection of distribution (OOD) samples. RankOOD constructs canonical rank orders using a 0–1 integer linear programming formulation guided by semantic similarity from probability mass functions (PMFs) of a pre-trained network, and optimises them via a Listwise Maximum Likelihood Estimation (ListMLE) objective. Extensive evaluations on CIFAR-10, CIFAR-100, and TinyImageNet demonstrate that RankOOD consistently ranks among the top two methods for near-OOD detection and the top three for far-OOD detection across 34 baselines. Notably, RankOOD achieves superior AUROC and FPR95 on the challenging TinyImageNet near-OOD benchmark, highlighting its ability to capture fine-grained semantic distinctions between ID and OOD samples.

Chapter 7

Conclusion

This thesis investigates content rating compliance in mobile application markets, an area that has received limited research attention despite rapid advances in generative artificial intelligence and multimodal vision–language technologies. Mobile applications have become integral to daily life across all age groups and sectors, including finance, healthcare, cybersecurity, and cyber-physical systems. However, the extent to which malicious actors exploit mobile platforms for monetisation through inappropriate or deceptive content practices has only recently begun to be examined.

The central research question of this thesis is:

> *How can content rating compliance in mobile applications be identified across different app markets, despite developer bias and legal variability among platforms, and how can an automated method be designed to centralise and enforce heterogeneous content rating schemes?*

To address this question, the thesis analyses the evolution of mobile app markets with respect to revenue-driven practices, content rating manipulation, and the fragmentation of rating frameworks. These factors introduce practical constraints imposed by developers and platform operators, often at the expense of user safety, particularly for minors. By identifying rating malpractices and proposing a centralised, automated content rating approach, this research aims to bridge the gap between legal vulnerability analysis and the practical detection of inappropriate mobile application content.

This chapter concludes the thesis by summarising its main contributions and consolidating the findings from the preceding studies. Further, the chapter outline with reflections on the broader implications of this work, highlighting the applicability of the presented techniques to other domains, including out-of-distribution detection. Finally, the chapter outlines potential directions for future research.

7.1 Summary and conclusion

This thesis presented four core contributions. The first three address challenges related to mobile application content appropriateness and compliance, with a focus on user safety. The final contribution extends the applicability of the proposed techniques to domains beyond mobile app markets. Collectively, these contributions provide a systematic analysis of vulnerabilities in existing content rating schemes and identify pathways toward more robust and automated content rating assignment.

7.1.1 Longitudinal Study of Apps via Multi-modal Majority Voting

In Chapter 3, we examined how mobile app metadata is highly dynamic, shifting in ways that complicate the task of tracking apps over long periods of time. Although the Google Play Store assigns each app a nominally unique app ID, our analysis shows that this identifier alone is insufficient for long-term re-identification. App IDs may change due to developer decisions, market-operator requirements, acquisitions, or deliberate strategic transformations. This instability highlights the need for more sophisticated cross-temporal app identification methods.

We proposed a novel, *multimodal majority-voting–based app similarity search methodology* capable of matching applications across two large-scale Google Play Store snapshots taken five years apart. Our approach integrates app icon features, textual semantics from descriptions, and TF-IDF representations of app and developer names. Through majority voting across these modalities, our system robustly identifies the closest counterpart of a 2018 app in the 2023 snapshot, even when app IDs change. This enables large-scale longitudinal studies of app behaviour and evolution-tasks previously infeasible with app ID–based matching alone.

Using this framework, we uncovered *eight distinct categories of app metamorphosis*. To quantify the impact of such transformations, we proposed a *Success Score* (SS) metric. Formally, SS reflects the extent to which an app's performance gains or falls behind the ecosystem's baseline expansion. Our results reveal several key insights. Re-branding emerges as the most successful metamorphosis strategy (average SS $\sim 21\%$), outperforming even top-tier, non-transformed apps by approximately 11%. Re-purposing also demonstrates above-baseline performance (SS $\sim 14\%$), illustrating that functional changes can become successful when executed carefully. In contrast, re-births, which require restarting under a new app ID, are the most challenging (SS $\sim -15\%$), reflecting the difficulty of rebuilding lost user bases.

Our analysis further uncovered systemic shifts in content ratings, a metamorphosis form extending beyond market strategy. We identified 504 apps whose content ratings shifted over five years, including cases where developers tightened ratings due to increased violence or mature themes, and cases where ratings were relaxed. This metamorphosis category remains critically under-explored in the literature, despite its significance for child protection, user safety, and regulatory compliance. Therefore opens an important research avenue into detecting content-rating

compliance in mobile markets.

Limitations: Our study focuses on intra-app evolution, analysing changes across successive versions of the same app. We do not address functionally similar applications developed by different authors (e.g., counterfeit apps), nor do we consider code-level similarity or repackaging detection. Existing work on repackaged, counterfeit, or malicious mobile applications typically relies on fine-grained analyses, such as bytecode similarity, control-flow graph comparison, or dynamic runtime analysis using one or more modalities. We argue that the proposed method is complementary and can be integrated into such frameworks to assist in detecting functional similarity or malicious modifications.

7.1.2 Contrastive Learning for Content Rating Prediction

Chapter 4 introduced a *supervised contrastive learning framework* for identifying content rating violations in mobile app markets. The proposed approach is grounded in the importance of multi-modality, jointly leveraging app creatives (i.e., icons and screenshots) and textual descriptions. To address content–style disparity in app metadata, the method separately integrates content and style encoders, trained using online and target networks updated via an exponentially moving average. These encoders learn visual and textual representations associated with content rating cues, such as cartoonish versus realistic violence. After representation learning, the ordinal nature of content ratings is exploited by training a ranking-based classifier using the ListMLE loss to predict app ratings.

Experimental results show that the proposed method outperforms state-of-the-art CLIP-based models, including fine-tuned variants, at least by 8%. Beyond performance gains, the model reveals two critical phenomena that pose risks to users: *malpractices*, where apps are assigned lower ratings than their content suggests, and *disguises*, where apps with correct ratings visually target a general or younger audience despite containing mature content. In the evaluated dataset, the model identified 71 apps ($\sim 17\%$ of verified cases) exhibiting rating malpractices, 32 apps showing disguising behaviour, and further found that 16.86% of violating apps were later removed from the store. Importantly, the analysis of 2,172 teacher-approved apps uncovered nine potential violations, highlighting risks to minors.

Overall, this work demonstrates that supervised contrastive learning enables robust learning of visual and textual cues relevant to content ratings. The proposed model is adaptable to different content rating schemes by training only the classifier and provides a practical tool for identifying risky apps, reducing manual review effort, and improving transparency and user safety in app markets.

Limitations: Prior work has demonstrated that contrastive learning objectives benefit from large batch sizes, as they effectively increase the number of negative samples available during training and thereby promote more discriminative representations [114, 236]. However, this

reliance on large batch sizes introduces practical challenges for model training, as it substantially increases computational and memory requirements. With increased GPU resources, the proposed method can be further scaled to enhance cross-modal alignment and representation quality, leading to improved separation among content rating classes.

7.1.3 CRD Reasoning with QwenSafe

Chapter 5 addresses a key limitation identified in the preceding work, where content rating prediction was performed without providing explanatory evidence. To overcome this, we extended our previous work by introducing **QwenSafe**, *a multimodal vision–language model built on the Qwen3 backbone, designed to identify content rating descriptors and generate explicit reasoning* grounded in app screenshots and textual descriptions. This shift from label-only prediction to evidence-based reasoning improves transparency and interpretability in automated content rating systems.

To support effective domain adaptation for content rating data, we proposed a novel metadata2CRD pipeline, which reformulates multimodal app data into a conversational training format aligned with a unified descriptor taxonomy. The resulting framework generalises across heterogeneous rating schemes and reduces dependence on region-specific or platform-specific rating algorithms. Experimental results demonstrate that QwenSafe substantially improves the detection of inappropriate content. Compared to the QwenSafe, the positive recall of the Qwen3 baseline decreased by 52.8% in binary classification and by 39% in impact-based multi-class classification. QwenSafe also achieved a 2.1% improvement in positive recall over Gemini, a large-scale multimodal model trained on substantially broader data, indicating strong generalisation in a safety-critical domain.

Beyond performance improvements, the analysis reveals under-reporting of content descriptors in major app marketplaces. In particular, the Android Play Store frequently omits descriptors that are visually or semantically present in the app content and even permitted under existing regional authorities. This finding underscores the limitations of current self-disclosure and review mechanisms and motivates the need for automated tools.

Centralising content rating schemes poses significant legal and social challenges. However, content rating descriptors offer a more viable basis for unification. This chapter presents a generic and automated framework for content descriptor identification that operates independently of region- and platform-specific rating systems, enabling a unified, scalable, and explainable approach to content rating verification across mobile app ecosystems.

Limitations: QwenSafe focuses on analysing app descriptions and screenshots available at submission time, which aligns well with scalable marketplace review workflows. However, some content rating descriptors, particularly those related to interactive elements, evolving user-generated content, or dynamic in-app experiences/divertissements may only manifest during

runtime. Extending the framework to incorporate dynamic signals could further strengthen its coverage.

7.1.4 Out-Of-Distribution Detection via RankOOD

Chapter 4 leveraged ListMLE loss to exploit the inherent ordinal structure of content rating classes. Chapter 6 extends this idea to out-of-distribution detection by leveraging implicit class ranking. Specifically, Chapter 6 introduced *RankOOD, a training-based OOD detection framework that utilises class-wise ranking structures* to distinguish in-distribution from out-of-distribution samples. By learning canonical class rankings and optimising a listwise objective based on the Plackett–Luce formulation, RankOOD enforces ranking consistency beyond top-1 predictions. The proposed RankOOD score explicitly penalises deviations from the expected ranking order, providing an effective OOD signal. Extensive evaluations against 34 baselines demonstrate that RankOOD consistently achieves top-tier performance, ranking within the top two methods for near-OOD detection and the top three for far-OOD detection. Notably, it outperforms existing rank-based approaches, including the strongest baseline CRAFT, with average FPR95 reductions of 4.21% in near-OOD and 7.51% in far-OOD settings while achieving SOTA FPR95 performance on TinyImageNet in near-OOD setting. These gains are achieved without architectural modifications or reliance on auxiliary outlier data.

This work shows that leveraging information from multiple ranks yields stronger OOD discrimination than relying solely on the top-ranked class, particularly as the number of classes increases. RankOOD thus provides a scalable and conceptually simple extension to rank-based OOD detection, highlighting the importance of structured class relationships for reliable OOD detection.

To better align RankOOD with the core research of the thesis, we note that the underlying principle of exploiting structured class relationships can be naturally extended to content rating compliance. In particular, content rating categories inherently follow an ordinal and semantically structured hierarchy (e.g., increasing levels of violence, maturity, or risk). By modelling these relationships through ranking-based objectives, deviations from expected descriptor or rating orderings can serve as indicators of potential non-compliance. For instance, an application whose learned representation induces inconsistent rankings across content descriptors (e.g., low predicted maturity despite strong high-risk cues) can be interpreted as exhibiting rating irregularities similar to the malpractices identified in earlier chapters. In this sense, RankOOD provides a complementary perspective: rather than detecting distributional anomalies at the top rank output prediction alone, it captures inconsistencies in the relational structure of predictions. This suggests a unified framework in which ranking-based signals are jointly leveraged for both out-of-distribution detection and content compliance verification, offering a novel approach to identifying subtle, adversarially manipulated app content.

Limitations: ListMLE optimises the likelihood of observing a complete ranking, which allows it to capture relative ordering information; however, it also introduces sensitivity to the exact ranking structure present in the training data. This sensitivity can lead to overfitting when the underlying differences between classes are small, yet a strict ordering is imposed. Such effects are more evident in high-cardinality datasets with highly semantically similar categories.

7.2 Future Work

While the proposed approaches demonstrate effectiveness in identifying inappropriate applications, they are currently limited to mobile app metadata, specifically screenshots and textual descriptions. Extending the framework to incorporate additional sources of app metadata remains an important direction for future work. One extension is the integration of other app metadata, such as user reviews and data safety declarations (**cf.** Figure 1.3). Prior work has shown that combining static and dynamic analyses of data safety declarations can effectively identify inconsistencies [316]. Other studies have leveraged mobile app user reviews to detect hate speech, violent content, and other harmful user experiences using large language models and natural language processing techniques [317, 318]. Hate speech detection has also been extended to multimodal data (text, image, and video) extracted from social media platforms, with various multimodal fusion strategies proposed, including early fusion [319], late fusion [320], and joint representation learning [321].

Another promising direction is the inclusion of video-based content, such as trailer videos and screen-recordings of mobile apps. Modeling video data can provide additional contextual cues about application functionality and expose information not captured in app store pages. Although mobile app video-based content analysis has not been extensively studied, prior research has addressed harmful video detection on social media platforms [322]. The recent emergence of multimodal large language models (MLLMs) capable of reasoning over text, images, and videos, such as Qwen3-VL [156], Gemini 2.5 Pro [49], and the GPT series [48] opens new opportunities for multimodal content classification, particularly in the context of video-centric digital platforms.

We utilized a contrastive learning approach to generate embeddings for mobile app metadata in Chapter 4. However, contrastive learning relies on negative examples, which increases the batch size and often requires hard sample mining, a procedure that introduces additional computational overhead and complexity during training. With the introduction of ViT [206], masked multimodal modeling (MMM) gained prominence, achieving strong performances. Previous MMM work such as FLAVA [128] and Unified-IO [323] use different fusion techniques to unified the modalities or treat them separately. It would be interesting to incorporate our findings regarding mobile app content domain into more recent such techniques.

Bibliography

[1] Google, "Google Play," https://play.google.com/store/games?hl=en, 2012.

[2] Apple, "App Store," https://www.apple.com/au/app-store/, 2008.

[3] 42matters, "Google Play Statistics and Trends 2025," https://42matters.com/google-play-statistics-and-trends, 2025.

[4] 42matters, "iOS Apple App Store Statistics and Trends 2025," https://42matters.com/ios-apple-app-store-statistics-and-trends, 2025.

[5] Ash Turner, "How Many Smartphones Are In The World? (2025)," https://www.bankmycell.com/blog/how-many-phones-are-in-the-world, 2025.

[6] Pew Research Center, "Mobile Fact Sheet, FACT SHEETS: TECH ADOPTION TRENDS," https://www.pewresearch.org/internet/fact-sheet/mobile/, 2024.

[7] Statista, "Smartphone penetration rate as share of the population in Singapore from 2020 to 2029," https://www.statista.com/statistics/625441/smartphone-user-penetration-in-singapore/, 2025.

[8] Statista, "Smartphone market in Australia - statistics & facts," https://www.statista.com/topics/4605/smartphone-market-in-australia/, 2025.

[9] Pew Research Center, "Teens, Social Media and Technology 2024," https://www.pewresearch.org/internet/2024/12/12/teens-social-media-and-technology-2024/, 2024.

[10] Common Sense, "The Common Sense Census: Media Use by Tweens and Teens, 2019," https://www.commonsensemedia.org/research/the-common-sense-census-media-use-by-tweens-and-teens-2019, 2019.

[11] Statista Research Department, "Number of active online banking users worldwide in 2020 with forecasts from 2021 to 2024, by region," https://www.statista.com/statistics/1228757/online-banking-users-worldwide/, 2025.

[12] Laura Ceci, "Most popular Android learning apps worldwide 2024, by downloads," https://www.statista.com/statistics/1497615/popular-education-apps-android-by-downloads/, 2025.

[13] Conor Stewart, "Penetration rate of online doctor consultations worldwide 2017-2029," https://www.statista.com/forecasts/1456734/online-doctor-consultations-penetration-rate-worldwide/, 2025.

[14] Aniello Cimitile, Fabio Martinelli, Francesco Mercaldo, et al., "Machine Learning Meets iOS Malware: Identifying Malicious Applications on Apple Environment.," in *ICISSP*, 2017, pp. 487–492.

[15] Google developer, "Google Play Protect," https://developers.google. com/android/play-protect, 2025.

[16] Andrea Caruso, *Forensic Analysis of Mobile Spyware: Investigating Security, Vulnerabilities, and Detection Challenges in Android and iOS Platforms*, Ph.D. thesis, Politecnico di Torino, 2024.

[17] Suranga Seneviratne, Aruna Seneviratne, Mohamed Ali Kaafar, Anirban Mahanti, and Prasant Mohapatra, "Spam mobile apps: Characteristics, detection, and in the wild analysis," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 1, pp. 1–29, 2017.

[18] Suranga Seneviratne, Aruna Seneviratne, Mohamed Ali Kaafar, Anirban Mahanti, and Prasant Mohapatra, "Early detection of spam mobile apps," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 949–959.

[19] Haoyu Wang, Yao Guo, Ziang Ma, and Xiangqun Chen, "Wukong: A scalable and accurate two-phase approach to android app clone detection," in *Proceedings of the 2015 international symposium on software testing and analysis*, 2015, pp. 71–82.

[20] Jonathan Crussell, Clint Gibler, and Hao Chen, "Attack of the clones: Detecting cloned applications on android markets," in *European Symposium on Research in Computer Security*. Springer, 2012, pp. 37–54.

[21] Jathushan Rajasegaran, Naveen Karunanayake, Ashanie Gunathillake, Suranga Seneviratne, and Guillaume Jourjon, "A multi-modal neural embeddings approach for detecting mobile counterfeit apps," in *The World Wide Web Conference*, 2019, pp. 3165–3171.

[22] Naveen Karunanayake, Jathushan Rajasegaran, Ashanie Gunathillake, Suranga Seneviratne, and Guillaume Jourjon, "A multi-modal neural embeddings approach for detecting mobile counterfeit apps: A case study on Google Play store," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 16–30, 2020.

[23] Hengshu Zhu, Hui Xiong, Yong Ge, and Enhong Chen, "Discovery of ranking fraud for mobile apps," *IEEE Transactions on knowledge and data engineering*, vol. 27, no. 1, pp. 74–87, 2014.

[24] Didi Surian, Suranga Seneviratne, Aruna Seneviratne, and Sanjay Chawla, "App miscategorization detection: A case study on google play," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1591–1604, 2017.

[25] Yangyu Hu, Haoyu Wang, Ren He, Li Li, Gareth Tyson, Ignacio Castro, Yao Guo, Lei Wu, and Guoai Xu, "Mobile app squatting," in *Proceedings of The Web Conference 2020*, 2020, pp. 1727–1738.

[26] Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt, "Third party tracking in the mobile ecosystem," in *Proceedings of the 10th ACM Conference on Web Science*, 2018, pp. 23–31.

[27] Suranga Seneviratne, Harini Kolamunna, and Aruna Seneviratne, "A measurement study of tracking in paid mobile applications," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2015, pp. 1–6.

[28] European General Data Protection Regulation, "General Data Protection Regulation GDPR.," https://gdpr-info.eu/, 2016.

[29] "CCPA. 2022. California Consumer Privacy Act (CCPA)," https://oag.ca.gov/privacy/ccpa. 2022/04/29., 2022.

[30] National Archives, "Children's online privacy protection rule," https://www.ecfr.gov/current/title-16/chapter-I/subchapter-C/part-312, 2022.

[31] Rishabh Khandelwal, Asmit Nayak, Paul Chung, and Kassem Fawaz, "Unpacking privacy labels: A measurement and developer perspective on google's data safety section," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2831–2848.

[32] Abhijith Athreya Mysore Gopinath, Vinayshekhar Bannihatti Kumar, Shomir Wilson, and Norman Sadeh, "Automatic section title generation to improve the readability of privacy policies," *USENIX SOUPS*, 2020.

[33] Mukund Srinath, Shomir Wilson, and C Lee Giles, "Privacy at scale: Introducing the PrivaSeer corpus of web privacy policies," *arXiv preprint arXiv:2004.11131*, 2020.

[34] Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N Cameron Russell, et al., "The creation and analysis of a website privacy policy corpus," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1330–1340.

[35] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G Shin, and Karl Aberer, "Polisis: Automated analysis and presentation of privacy policies using deep learning," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 531–548.

[36] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman M Sadeh, Steven M Bellovin, and Joel R Reidenberg, "Automated Analysis of Privacy Requirements for Mobile Apps.," in *NDSS*, 2017, vol. 2, pp. 1–4.

[37] Google, "Apps & Games content ratings on Google Play," https://support.google.com/googleplay/answer/6209544?hl=en, 2025.

[38] Transport Regional Development Department of Infrastructure and Communication., "The advisory categories for films and computer games.," https://www.classification.gov.au/classification-ratings/what-are-ratings, 2025.

[39] International Age Rating Coalition., "How IARC works.," https://www. globalratings.com/how-iarc-works.aspx, 2025.

[40] Interactive Software Federation of Europe (ISFE), "PEGI-Pan-European Game Information," https://web.archive.org/web/20160213014044/http://www.pegi.info/en/index/id/952, 2003.

[41] Entertainment Software Rating Board, "Rating Guide," https://www.esrb.org/ratings-guide/, 1994.

[42] Unterhaltungssoftware Selbstkontrolle., "SK age categories.," https://usk.de/en/the-usk/faqs/age-categories/, 2025.

[43] Game Rating and Administration Committee, "Game Rating and Administration Committee.," https://www.grac.or.kr/english/, 2013.

[44] Apple Inc., "Age ratings values and definitions.," https://developer.apple.com/help/app-store-connect/reference/age-ratings-values-and-definitions, 2025.

[45] Apple Inc., "Set an app age rating.," https://developer.apple.com/help/app-store-connect/manage-app-information/set-an-app-age-rating, 2025.

[46] Canadian centre for child protection, "Reviewing the Enforcement of App Age Ratings in Apple's App Store and Google Play," https://content.c3p.ca/pdfs/C3P_AppAgeRatingReport_en.pdf, 2022.

[47] Ruoxi Sun, Minhui Xue, Gareth Tyson, Shuo Wang, Seyit Camtepe, and Surya Nepal, "Not seen, not heard in the digital world! measuring privacy practices in children's apps," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2166–2177.

[48] OpenAI, "ChatGPT: October 2023 version," https://chat.openai.com/, 2023.

[49] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al., "Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities," *arXiv preprint arXiv:2507.06261*, 2025.

[50] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al., "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[51] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International conference on machine learning*. PMLR, 2021, pp. 4904–4916.

[52] Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao, "Llava-med: Training a large language-and-vision assistant for biomedicine in one day," *Advances in Neural Information Processing Systems*, vol. 36, pp. 28541–28564, 2023.

[53] Ying Chen, Heng Xu, Yilu Zhou, and Sencun Zhu, "Is this app safe for children? A comparison study of maturity ratings on Android and iOS applications," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 201–212.

[54] Chenyu Zhou, Xian Zhan, Linlin Li, and Yepang Liu, "Automatic Maturity Rating for Android Apps," in *Proceedings of the 13th Asia-Pacific Symposium on Internetware*, 2022, pp. 16–27.

[55] Bing Hu, Bin Liu, Neil Zhenqiang Gong, Deguang Kong, and Hongxia Jin, "Protecting your children from inappropriate content in mobile apps: An automatic maturity rating framework," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1111–1120.

[56] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li, "Listwise approach to learning to rank: theory and algorithm," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1192–1199.

[57] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[58] "1998. Children's Online Privacy Protection Rule ("COPPA")," https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reformproceedings/childrens-online-privacy-protection-rule, 1998.

[59] "Children's Internet Protection Act.," http://ifea.net/cipa.pdf, 2000.

[60] "2017. Communications Decency Act of 1996.(CDA)," http://www.cybertelecom.org/cda/cda.htm., 2017.

[61] Motion Picture Association, "Film Rating," https://www.motionpictures.org/film-ratings/, 1968.

[62] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shomir Wilson, Norman Sadeh, Steven Bellovin, and Joel Reidenberg, "Automated analysis of privacy requirements for mobile apps," in *2016 AAAI Fall Symposium Series*, 2016.

[63] Irwin Reyes, Primal Wijesekera, Joel Reardon, Amit Elazari Bar On, Abbas Razaghpanah, Narseo Vallina-Rodriguez, Serge Egelman, et al., ""Won't somebody think of the children?" examining COPPA compliance at scale," in *The 18th Privacy Enhancing Technologies Symposium (PETS 2018)*, 2018.

[64] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie, "{PolicyLint}: Investigating Internal Privacy Policy Contradictions on Google Play," in *28th USENIX security symposium (USENIX security 19)*, 2019, pp. 585–602.

[65] Charles Weir, Ben Hermann, and Sascha Fahl, "From needs to actions to secure apps? the effect of requirements and developer practices on app security," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 289–305.

[66] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E Hassan, "What do mobile app users complain about?," *IEEE software*, vol. 32, no. 3, pp. 70–77, 2014.

[67] Gian Luca Scoccia, Stefano Ruberto, Ivano Malavolta, Marco Autili, and Paola Inverardi, "An investigation into Android run-time permissions from the end users' perspective," in *Proceedings of the 5th international conference on mobile software engineering and systems*, 2018, pp. 45–55.

[68] Martin Degeling, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz, "We value your privacy... now take some cookies: Measuring the GDPR's impact on web privacy," *arXiv preprint arXiv:1808.05096*, 2018.

[69] Vaibhav Rastogi, Rui Shao, Yan Chen, Xiang Pan, Shihong Zou, and Ryan D Riley, "Are these Ads Safe: Detecting Hidden Attacks through the Mobile App-Web Interfaces.," in *NDSS*, 2016.

[70] Gong Chen, Wei Meng, and John Copeland, "Revisiting mobile advertising threats with MAdLife," in *The World Wide Web Conference*, 2019, pp. 207–217.

[71] Minxing Liu, Haoyu Wang, Yao Guo, and Jason Hong, "Identifying and analyzing the privacy of apps for kids," in *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, 2016, pp. 105–110.

[72] Bing Hu, Bin Liu, Neil Zhenqiang Gong, Deguang Kong, and Hongxia Jin, "Protecting your children from inappropriate content in mobile apps: An automatic maturity rating framework," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1111–1120.

[73] "Developer Content Policy," https://developer.apple.com/app-store/review/guidelines/, Accessed: 2024-12-05.

[74] "Developer Content Policy," https://support.google.com/googleplay/android-developer/answer/9898843?hl=en, Accessed: 2024-12-05.

[75] Lawrence A Kutner, Cheryl K Olson, Dorothy E Warner, and Sarah M Hertzog, "Parents' and sons' perspectives on video game play: A qualitative study," *Journal of Adolescent Research*, vol. 23, no. 1, pp. 76–96, 2008.

[76] Monica K Miller, "Exploring the relationship between video game ratings implementation and changes in game content as represented by game magazines," *Politics & Policy*, vol. 38, no. 4, pp. 705–735, 2010.

[77] Leon Y Xiao, "Shopping around for loot box presence warning labels: unsatisfactory compliance on Epic, Nintendo, Sony, and Microsoft platforms," *ACM Games: Research and Practice*, vol. 1, no. 4, pp. 1–25, 2023.

[78] Elizabeth Handsley and Wayne Warburton, "'Material likely to harm or disturb them': testing the alignment between film and game classification decisions and psychological research evidence," *Psychiatry, psychology and law*, vol. 29, no. 1, pp. 68–92, 2022.

[79] Daniel L King, Paul H Delfabbro, Jeffrey L Derevensky, and Mark D Griffiths, "A review of Australian classification practices for commercial video games featuring simulated gambling," *International Gambling Studies*, vol. 12, no. 2, pp. 231–242, 2012.

[80] Steven Conway and Laura M Crawford, "Games for Grown-Ups?: An Historical Account of the Australian Classification System," in *Video Game Policy*, pp. 85–97. Routledge, 2015.

[81] Trung Tin Nguyen, Michael Backes, Ninja Marnau, and Ben Stock, "Share First, Ask Later (or Never?)-Studying Violations of GDPR's Explicit Consent in Android Apps," in *USENIX Security Symposium*, 2021.

[82] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman, "Actions speak louder than words:{Entity-Sensitive} privacy policy and data flow analysis with {PoliCheck}," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 985–1002.

[83] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé Bissyandé, and Jacques Klein, "Maddroid: Characterizing and detecting devious ad contents for android apps," in *Proceedings of The Web Conference 2020*, 2020, pp. 1715–1726.

[84] Konrad Kollnig, Pierre Dewitte, Max Van Kleek, Ge Wang, Daniel Omeiza, Helena Webb, and Nigel Shadbolt, "A Fait Accompli? An Empirical Study into the Absence of Consent to {Third-Party} Tracking in Android Apps," in *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, 2021, pp. 181–196.

[85] Trung Tin Nguyen, Michael Backes, and Ben Stock, "Freely given consent? studying consent notice of third-party tracking and its violations of gdpr in android apps," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2369–2383.

[86] Iskander Sanchez-Rola, Matteo Dell'Amico, Platon Kotzias, Davide Balzarotti, Leyla Bilge, Pierre-Antoine Vervier, and Igor Santos, "Can i opt out yet? gdpr and the global illusion of cookie control," in *Proceedings of the 2019 ACM Asia conference on computer and communications security*, 2019, pp. 340–351.

[87] Célestin Matte, Nataliia Bielova, and Cristiana Santos, "Do cookie banners respect my choice?: Measuring legal compliance of banners from iab europe's transparency and consent framework," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 791–809.

[88] "Media Week," https://www.mediaweek.com.au/australian-digital-advertising-grows-13 -billion-in-2021/, 2021.

[89] Michael Backes, Sven Bugiel, and Erik Derr, "Reliable third-party library detection in android and its security applications," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 356–367.

[90] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen, "Libradar: fast and accurate detection of third-party libraries in android apps," in *Proceedings of the 38th international conference on software engineering companion*, 2016, pp. 653–656.

[91] Jonathan Crussell, Ryan Stevens, and Hao Chen, "Madfraud: Investigating ad fraud in android applications," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014, pp. 123–134.

[92] Feng Dong, Haoyu Wang, Li Li, Yao Guo, Tegawendé F Bissyandé, Tianming Liu, Guoai Xu, and Jacques Klein, "Frauddroid: Automated ad fraud detection for android apps," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 257–268.

[93] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes, "Keep me updated: An empirical study of third-party library updatability on android," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 2187–2200.

[94] Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, and David Wagner, "Addroid: Privilege separation for applications and advertisers in android," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, 2012, pp. 71–72.

[95] "Google Play Store malware targets porn ads at millions of kids.," http://www.itpro.co.uk/malware/30294/google-play-store-malware-targets-porn-ads-at-millions-of-kids.

[96] "Malware Displaying Porn Ads Discovered in Game Apps on Google Play.," https://research.checkpoint.com/malware-displaying-porn-ads-discovered-in-game-apps-on-google-play.

[97] Europian Union, "2002/58 - EN - eprivacy directive - EUR-Lex - European Union," https://eur-lex.europa.eu/eli/dir/2002/58/oj/eng.

[98] Zheyuan Gu, Gaopeng Gou, Chang Liu, Chen Yang, Xiyuan Zhang, Zhen Li, and Gang Xiong, "Let gambling hide nowhere: Detecting illegal mobile gambling apps via heterogeneous graph-based encrypted traffic analysis," *Computer Networks*, vol. 243, pp. 110278, 2024.

[99] Arthur C Serra, Paulo Renato C Mendes, Pedro VA de Freitas, Antonio José G Busson, Álan LV Guedes, and Sérgio Colcher, "Should i see or should i go: Automatic detection of sensitive media in messaging apps," in *Proceedings of the Brazilian Symposium on Multimedia and the Web*, 2021, pp. 229–236.

[100] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[101] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[102] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[103] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov, "Unsupervised cross-lingual representation learning at scale," *arXiv preprint arXiv:1911.02116*, 2019.

[104] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran, "Self-supervised learning by cross-modal audio-video clustering," *Advances in neural information processing systems*, vol. 33, pp. 9758–9770, 2020.

[105] Yuki Asano, Mandela Patrick, Christian Rupprecht, and Andrea Vedaldi, "Labelling unlabelled videos from scratch with multi-modal self-supervision," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4660–4671, 2020.

[106] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi, "Self-labelling via simultaneous clustering and representation learning," *arXiv preprint arXiv:1911.05371*, 2019.

[107] Di Hu, Feiping Nie, and Xuelong Li, "Deep multimodal clustering for unsupervised audiovisual learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9248–9257.

[108] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al., "Improving language understanding by generative pre-training," *https://www.mikecaptain.com/resources/pdf/GPT-1.pdf*, 2018.

[109] Peizhao Li, Jiuxiang Gu, Jason Kuen, Vlad I Morariu, Handong Zhao, Rajiv Jain, Varun Manjunatha, and Hongfu Liu, "Selfdoc: Self-supervised document representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5652–5660.

[110] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid, "Videobert: A joint model for video and language representation learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7464–7473.

[111] Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao, "Simvlm: Simple visual language model pretraining with weak supervision," *arXiv preprint arXiv:2108.10904*, 2021.

[112] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu, "Bert-attack: Adversarial attack against bert using bert," *arXiv preprint arXiv:2004.09984*, 2020.

[113] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16000–16009.

[114] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[115] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He, "Improved baselines with momentum contrastive learning," *arXiv preprint arXiv:2003.04297*, 2020.

[116] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[117] Yonglong Tian, Dilip Krishnan, and Phillip Isola, "Contrastive multiview coding," in *European conference on computer vision*. Springer, 2020, pp. 776–794.

[118] Jean-Baptiste Alayrac, Adria Recasens, Rosalia Schneider, Relja Arandjelović, Jason Ramapuram, Jeffrey De Fauw, Lucas Smaira, Sander Dieleman, and Andrew Zisserman, "Self-supervised multimodal versatile networks," *Advances in neural information processing systems*, vol. 33, pp. 25–37, 2020.

[119] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong, "Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text," *Advances in neural information processing systems*, vol. 34, pp. 24206–24221, 2021.

[120] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel, "Audioclip: Extending clip to image, text and audio," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 976–980.

[121] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer, "Videoclip: Contrastive pre-training for zero-shot video-text understanding," *arXiv preprint arXiv:2109.14084*, 2021.

[122] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li, "Pointclip: Point cloud understanding by clip," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8552–8562.

[123] Michael Tschannen, Basil Mustafa, and Neil Houlsby, "Image-andlanguage understanding from pixels only," *arXiv preprint arXiv:2212.08045*, vol. 3, 2022.

[124] Yanghao Li, Haoqi Fan, Ronghang Hu, Christoph Feichtenhofer, and Kaiming He, "Scaling language-image pre-training via masking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 23390–23400.

[125] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie, "Slip: Self-supervision meets language-image pre-training," in *European conference on computer vision*. Springer, 2022, pp. 529–544.

[126] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu, "Uniter: Universal image-text representation learning," in *European conference on computer vision*. Springer, 2020, pp. 104–120.

[127] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi, "Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation," in *International conference on machine learning*. PMLR, 2022, pp. 12888–12900.

[128] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela, "Flava: A foundational language and vision alignment model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15638–15650.

[129] Florian Bordes, Randall Balestriero, and Pascal Vincent, "Towards democratizing joint-embedding self-supervised learning," *arXiv preprint arXiv:2303.01986*, 2023.

[130] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D Manning, and Curtis P Langlotz, "Contrastive learning of medical visual representations from paired images and text," *arXiv preprint arXiv:2010.00747*, 2020.

[131] Shih-Cheng Huang, Liyue Shen, Matthew P Lungren, and Serena Yeung, "Gloria: A multimodal global-local representation learning framework for label-efficient medical image recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3942–3951.

[132] Zifeng Wang, Zhenbang Wu, Dinesh Agarwal, and Jimeng Sun, "Medclip: Contrastive learning from unpaired medical images and text," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, 2022, vol. 2022, p. 3876.

[133] Aiham Taleb, Matthias Kirchler, Remo Monti, and Christoph Lippert, "Contig: Self-supervised multimodal contrastive learning for medical imaging with genetics," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 20908–20921.

[134] Yuxing Chen and Lorenzo Bruzzone, "Self-supervised SAR-optical data fusion of Sentinel-1/-2 images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–11, 2021.

[135] Antonio Montanaro, Diego Valsesia, Giulia Fracastoro, and Enrico Magli, "Semi-supervised learning for joint SAR and multispectral land cover classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[136] Sudipan Saha, Patrick Ebel, and Xiao Xiang Zhu, "Self-supervised multisensor change detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–10, 2021.

[137] Kanishk Jain, Varun Chhangani, Amogh Tiwari, K Madhava Krishna, and Vineet Gandhi, "Ground then navigate: Language-guided navigation in dynamic scenes," *arXiv preprint arXiv:2209.11972*, 2022.

[138] Runnan Chen, Youquan Liu, Lingdong Kong, Xinge Zhu, Yuexin Ma, Yikang Li, Yuenan Hou, Yu Qiao, and Wenping Wang, "Clip2scene: Towards label-efficient 3d scene understanding by clip," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7020–7030.

[139] Pratyay Banerjee, Shweti Mahajan, Kushal Arora, Chitta Baral, and Oriana Riva, "Lexi: Self-supervised learning of the ui language," in *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022, pp. 6992–7007.

[140] Zhongxiang Fu, Buqing Cao, Shanpeng Liu, Qian Peng, Zhenlian Peng, Min Shi, and Shangli Liu, "SMAR: self-supervised mobile application recommendation based on graph convolutional networks," *International Journal of Web Information Systems*, vol. 20, no. 5, pp. 520–536, 2024.

[141] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Jieqi Kang, et al., "Self-supervised learning for large-scale item recommendations," in *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, pp. 4321–4330.

[142] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer, "Sigmoid loss for language image pre-training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11975–11986.

[143] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao, "Eva-clip: Improved training techniques for clip at scale," *arXiv preprint arXiv:2303.15389*, 2023.

[144] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," *arXiv preprint arXiv:2203.03605*, 2022.

[145] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al., "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[146] Gen Luo, Yiyi Zhou, Yuxin Zhang, Xiawu Zheng, Xiaoshuai Sun, and Rongrong Ji, "Feast your eyes: Mixture-of-resolution adaptation for multimodal large language models," *arXiv preprint arXiv:2403.03003*, 2024.

[147] Yanwei Li, Yuechen Zhang, Chengyao Wang, Zhisheng Zhong, Yixin Chen, Ruihang Chu, Shaoteng Liu, and Jiaya Jia, "Mini-gemini: Mining the potential of multi-modality vision language models," *arXiv preprint arXiv:2403.18814*, 2024.

[148] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al., "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[149] Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al., "Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality," *See https://vicuna. lmsys. org (accessed 14 April 2023)*, vol. 2, no. 3, pp. 6, 2023.

[150] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al., "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[151] Bin Lin, Zhenyu Tang, Yang Ye, Jiaxi Cui, Bin Zhu, Peng Jin, Junwu Zhang, Munan Ning, and Li Yuan, "MoE-LLaVA: Mixture of Experts for Large Vision-Language Models," *arXiv preprint arXiv:2401.15947*, 2024.

[152] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al., "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23716–23736, 2022.

[153] Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, and Fei Huang, "mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2024, pp. 13040–13051.

[154] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee, "Visual Instruction Tuning," 2023.

[155] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *International conference on machine learning*. PMLR, 2023, pp. 19730–19742.

[156] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu, "Qwen3-VL Technical Report," *arXiv preprint arXiv:2511.21631*, 2025.

[157] Tengchao Lv, Yupan Huang, Jingye Chen, Yuzhong Zhao, Yilin Jia, Lei Cui, Shuming Ma, Yaoyao Chang, Shaohan Huang, Wenhui Wang, et al., "Kosmos-2.5: A multimodal literate model," *arXiv preprint arXiv:2309.11419*, 2023.

[158] Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al., "Cogagent: A visual language model for gui agents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 14281–14290.

[159] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang, "An embodied generalist agent in 3d world," *arXiv preprint arXiv:2311.12871*, 2023.

[160] Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Yuhao Dan, Chenlin Zhao, Guohai Xu, Chenliang Li, Junfeng Tian, et al., "mplug-docowl: Modularized multimodal large language model for document understanding," *arXiv preprint arXiv:2307.02499*, 2023.

[161] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg, "Modeling Context in Referring Expressions," in *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, Eds., Cham, 2016, pp. 69–85, Springer International Publishing.

[162] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee, "LLaVA-NeXT: Improved reasoning, OCR, and world knowledge," January 2024.

[163] Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang, "Mobile-agent: Autonomous multi-modal mobile device agent with visual perception," *arXiv preprint arXiv:2401.16158*, 2024.

[164] Chi Zhang, Zhao Yang, Jiaxuan Liu, Yanda Li, Yucheng Han, Xin Chen, Zebiao Huang, Bin Fu, and Gang Yu, "Appagent: Multimodal agents as smartphone users," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025, pp. 1–20.

[165] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun, "MiniCPM-V: A GPT-4V Level MLLM on Your Phone," 2024.

[166] Xiangxiang Chu, Limeng Qiao, Xinyang Lin, Shuang Xu, Yang Yang, Yiming Hu, Fei Wei, Xinyu Zhang, Bo Zhang, Xiaolin Wei, et al., "Mobilevlm: A fast, reproducible and strong vision language assistant for mobile devices," *arXiv preprint arXiv:2312.16886*, vol. 2, no. 6, pp. 7, 2023.

[167] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, et al., "Minicpm-v: A gpt-4v level mllm on your phone," *arXiv preprint arXiv:2408.01800*, 2024.

[168] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin, "Qwen2.5-VL Technical Report," *arXiv preprint arXiv:2502.13923*, 2025.

[169] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine, "The hateful memes challenge: Detecting hate speech in multimodal memes," *Advances in neural information processing systems*, vol. 33, pp. 2611–2624, 2020.

[170] Rui Cao, Ming Shan Hee, Adriel Kuek, Wen-Haw Chong, Roy Ka-Wei Lee, and Jing Jiang, "Procap: Leveraging a frozen vision-language model for hateful meme detection," in *Proceedings of the 31st ACM international conference on multimedia*, 2023, pp. 5244–5252.

[171] Shraman Pramanick, Shivam Sharma, Dimitar Dimitrov, Md Shad Akhtar, Preslav Nakov, and Tanmoy Chakraborty, "MOMENTA: A multimodal framework for detecting harmful memes and their targets," *arXiv preprint arXiv:2109.05184*, 2021.

[172] Keyan Guo, Ayush Utkarsh, Wenbo Ding, Isabelle Ondracek, Ziming Zhao, Guo Freeman, Nishant Vishwamitra, and Hongxin Hu, "Moderating Illicit Online Image Promotion for Unsafe User Generated Content Games Using Large {Vision-Language} Models," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 5787–5804.

[173] Statista Inc., "Number of available applications in the Google Play Store from December 2009 to March 2023," https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/, 2023, Accessed on May, 2023.

[174] Statista Inc., "Number of apps available in leading app stores as of 3rd quarter 2022," https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, 2022, Accessed on May, 2023.

[175] TechCrunch, "Short video service Musical.ly is merging into sister app TikTok," https://techcrunch.com/2018/08/02/musically-tiktok/, 2018, Accessed on May, 2023.

[176] Nicolas Viennot, Edward Garcia, and Jason Nieh, "A measurement study of google play," in *The 2014 ACM international conference on Measurement and modeling of computer systems*, 2014, pp. 221–233.

[177] Haoyu Wang, Zhe Liu, Jingyue Liang, Narseo Vallina-Rodriguez, Yao Guo, Li Li, Juan Tapiador, Jingcun Cao, and Guoai Xu, "Beyond google play: A large-scale comparative study of chinese android app markets," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 293–307.

[178] Kai Chen, Xueqiang Wang, Yi Chen, Peng Wang, Yeonjoon Lee, XiaoFeng Wang, Bin Ma, Aohui Wang, Yingjun Zhang, and Wei Zou, "Following devil's footprints: Cross-platform analysis of potentially harmful libraries on android and ios," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 357–376.

[179] Oliviero Riganelli, Ionut Daniel Fagadau, Daniela Micucci, and Leonardo Mariani, "Proactive libraries: enforcing correct behaviors in Android apps," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, 2022, pp. 159–163.

[180] Bozhi Wu, Sen Chen, Cuiyun Gao, Lingling Fan, Yang Liu, Weiping Wen, and Michael R Lyu, "Why an android app is classified as malware: Toward malware classification interpretation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 2, pp. 1–29, 2021.

[181] Xiaolei Wang, Yuexiang Yang, and Sencun Zhu, "Automated hybrid analysis of android malware through augmenting fuzzing with forced execution," *IEEE Transactions on Mobile Computing*, vol. 18, no. 12, pp. 2768–2782, 2018.

[182] Michael Grace, Yajin Zhou, Qiang Zhang, Shihong Zou, and Xuxian Jiang, "Riskranker: scalable and accurate zero-day android malware detection," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, 2012, pp. 281–294.

[183] Asaf Shabtai, Uri Kanonov, Yuval Elovici, Chanan Glezer, and Yael Weiss, ""Andromaly": a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 161–190, 2012.

[184] Dong-Jie Wu, Ching-Hao Mao, Te-En Wei, Hahn-Ming Lee, and Kuo-Ping Wu, "Droidmat: Android malware detection through manifest and api calls tracing," in *2012 Seventh Asia joint conference on information security*. IEEE, 2012, pp. 62–69.

[185] Zhenlong Yuan, Yongqiang Lu, Zhaoguo Wang, and Yibo Xue, "Droid-sec: deep learning in android malware detection," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 371–372.

[186] Muhammad Ikram, Narseo Vallina-Rodriguez, Suranga Seneviratne, Mohamed Ali Kaafar, and Vern Paxson, "An analysis of the privacy and security risks of android vpn permission-enabled apps," in *Proceedings of the 2016 internet measurement conference*, 2016, pp. 349–364.

[187] Kaspar Hageman, Álvaro Feal, Julien Gamba, Aniketh Girish, Jakob Bleier, Martina Lindorfer, Juan Tapiador, and Narseo Vallina-Rodriguez, "Mixed Signals: Analyzing Software Attribution Challenges in the Android Ecosystem," *IEEE Transactions on Software Engineering*, 2023.

[188] Rahul Potharaju, Mizanur Rahman, and Bogdan Carbunar, "A longitudinal study of google play," *IEEE Transactions on computational social systems*, vol. 4, no. 3, pp. 135–149, 2017.

[189] Andreas Möller, Florian Michahelles, Stefan Diewald, Luis Roalter, and Matthias Kranz, "Update behavior in app markets and security implications: A case study in google play," in *Research in the Large, LARGE 3.0: 21/09/2012-21/09/2012*, 2012, pp. 3–6.

[190] Vincent F Taylor and Ivan Martinovic, "To update or not to update: Insights from a two-year study of android app evolution," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 2017, pp. 45–57.

[191] Md Ahasanuzzaman, Safwat Hassan, Cor-Paul Bezemer, and Ahmed E Hassan, "A longitudinal study of popular ad libraries in the Google Play Store," *Empirical Software Engineering*, vol. 25, pp. 824–858, 2020.

[192] Haoyu Wang, Hao Li, and Yao Guo, "Understanding the evolution of mobile app ecosystems: A longitudinal measurement study of google play," in *The World Wide Web Conference*, 2019, pp. 1988–1999.

[193] Tong Li, Yali Fan, Yong Li, Sasu Tarkoma, and Pan Hui, "Understanding the long-term evolution of mobile app usage," *IEEE Transactions on Mobile Computing*, 2021.

[194] Anushruti Vagrani, Niraj Kumar, and P Vigneswara Ilavarasan, "Decline in mobile application life cycle," *Procedia Computer Science*, vol. 122, pp. 957–964, 2017.

[195] Steve Hanna, Ling Huang, Edward Wu, Saung Li, Charles Chen, and Dawn Song, "Juxtapp: A scalable system for detecting code reuse among android applications," in *Detection of Intrusions and Malware, and Vulnerability Assessment: 9th International Conference, DIMVA 2012, Heraklion, Crete, Greece, July 26-27, 2012, Revised Selected Papers 9*. Springer, 2013, pp. 62–81.

[196] Wu Zhou, Yajin Zhou, Xuxian Jiang, and Peng Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, 2012, pp. 317–326.

[197] Fangfang Zhang, Heqing Huang, Sencun Zhu, Dinghao Wu, and Peng Liu, "ViewDroid: Towards obfuscation-resilient mobile application repackaging detection," in *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, 2014, pp. 25–36.

[198] MA Rahim Khan, RC Tripathi, and Ajit Kumar, "Repacked android application detection using image similarity," *Nexo Revista Científica*, vol. 33, no. 01, pp. 190–199, 2020.

[199] Quanlong Guan, Heqing Huang, Weiqi Luo, and Sencun Zhu, "Semantics-based repackaging detection for mobile apps," in *Engineering Secure Software and Systems: 8th International Symposium, ESSoS 2016, London, UK, April 6–8, 2016. Proceedings 8*. Springer, 2016, pp. 89–105.

[200] Su Mon Kywe, Yingjiu Li, Robert H Deng, and Jason Hong, "Detecting camouflaged applications on mobile application markets," in *Information Security and Cryptology-ICISC 2014: 17th International Conference, Seoul, South Korea, December 3-5, 2014, Revised Selected Papers 17*. Springer, 2015, pp. 241–254.

[201] Wu Zhou, Yajin Zhou, Michael Grace, Xuxian Jiang, and Shihong Zou, "Fast, scalable detection of" piggybacked" mobile applications," in *Proceedings of the third ACM conference on Data and application security and privacy*, 2013, pp. 185–196.

[202] Aisha Ali-Gombe, Irfan Ahmed, Golden G Richard III, and Vassil Roussev, "Opseq: Android malware fingerprinting," in *Proceedings of the 5th Program Protection and Reverse Engineering Workshop*, 2015, pp. 1–12.

[203] Wu Zhou, Xinwen Zhang, and Xuxian Jiang, "AppInk: watermarking android apps for repackaging deterrence," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*, 2013, pp. 1–12.

[204] Didi Surian, Suranga Seneviratne, Aruna Seneviratne, and Sanjay Chawla, "App miscategorization detection: A case study on google play," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1591–1604, 2017.

[205] Yingying Deng, Fan Tang, Weiming Dong, Chongyang Ma, Xingjia Pan, Lei Wang, and Changsheng Xu, "Stytr2: Image style transfer with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11326–11336.

[206] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[207] Yova Kementchedjhieva and Ilias Chalkidis, "An Exploration of Encoder-Decoder Approaches to Multi-Label Classification for Legal and Biomedical Text," *arXiv preprint arXiv:2305.05627*, 2023.

[208] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu, "Mpnet: Masked and permuted pre-training for language understanding," *Advances in Neural Information Processing Systems*, vol. 33, pp. 16857–16867, 2020.

[209] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[210] Jeff Johnson, Matthijs Douze, and Hervé Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[211] Fernando Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," https://github.com/fmfn/BayesianOptimization, 2014–.

[212] Ben Popper, "Google announces over 2 billion monthly active devices on Android," https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users.

[213] Ash Turner, "The Rise of Android: Why is Android Successful?," https://www.bankmycell.com/blog/how-many-android-users-are-there.

[214] Sam Nguyen, "Android's Latest Statistics 2024: How Many People Have Androids?," https://avada.io/articles/how-many-people-have-androids/, September 06, 2023.

[215] "Insights into the 2.3 Billion Android Smartphones in Use Around the World," https://newzoo.com/resources/blog/insights-into-the-2-3-billion-android-smartphones-in-use-around-the-world, January 13, 2018.

[216] Peak Frameworks Team, "Unlocking the Power of CAGR: A Key Metric for Investment Evaluation," https://www.peakframeworks.com/post/cagr-key-metric#: :text=Conclusion,in

[217] Jason Fernando, "Compound Annual Growth Rate (CAGR) Formula and Calculation (2024)," https://www.investopedia.com/terms/c/cagr.asp.

[218] Mark, "What is Compound Annual Growth Rate?(2023)," https://www.capitalcitytraining.com/knowledge/compound-annual-growth-rate-cagr/.

[219] "Uno (video game)," https://en.wikipedia.org/wiki/Uno_(video_game)#Gameloft_versions, 2022, Accessed on May, 2023.

[220] "Flickr," https://en.wikipedia.org/wiki/Flickr 2022, Accessed on May, 2023.

[221] "Transfer app to a different developer account," https://support.google.com/googleplay/android-developer/answer/6230247?hl=en, 2023, Accessed on May, 2023.

[222] Subway Surfers Wiki, "Kiloo - Subway Surfers Wiki," https://subwaysurf.fandom.com/wiki/Kiloo, Accessed on May, 2023.

[223] 42matters, "Google Play Statistics and Trends 2024," https://42matters.com/google-play-statistics-and-trends.

[224] Billy Lau, Jiexin Zhang, Alastair R Bereford, Daniel Thomas, and René Mayrhofer, "Uraniborg's device preloaded app risks scoring metrics," *Institute of Networks and Security: Linz, Austria*, 2020.

[225] Ash Turner, "Number Of Smartphone Users Worldwide (Billions)," https://www.bankmycell.com/blog/how-many-phones-are-in-the-world#part-1, 2024.

[226] "Build Teacher Approved apps," https://play.google.com/console/about/programs/teacherapproved/, Accessed: 2024-08-07.

[227] Qian Luo, Jiajia Liu, Jiadai Wang, Yawen Tan, Yurui Cao, and Nei Kato, "Automatic content inspection and forensics for children android apps," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7123–7134, 2020.

[228] Qustodio, "Qustodio releases 2023 Annual Report, Born connected: The rise of the AI generation," https://static.qustodio.com/public-site/uploads/2024/01/19122535/ADR_2023-24_EN.pdf, 2023.

[229] "Requirements related to content ratings for apps, games and the ads served on both," https://support.google.com/googleplay/android-developer/answer/9859655, Accessed: 2024-12-05.

[230] "Bingo Mania - Light Bingo Game," https://play.google.com/store/apps/details?id=com.lunavalley.bingo&hl=en, Accessed: 2025-08-05.

[231] "Cute cats app jigsaw puzzles," https://play.google.com/store/apps/details?id=com.sprpuzzle.sprpzzlgms&hl=en, Accessed: 2025-08-05.

[232] Federal Trade Commission, "Mobile apps for kids: Current privacy disclosures are disappointing," *Washington, DC. RetrievedAugust*, vol. 21, pp. 2022, 2012.

[233] Trung Tin Nguyen, Duc Cuong Nguyen, Michael Schilling, Gang Wang, and Michael Backes, "Measuring user perception for detecting unexpected access to sensitive resource in mobile apps," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 578–592.

[234] Ilaria Liccardi, Monica Bulger, Hal Abelson, Daniel J Weitzner, and Wendy Mackay, "Can apps play by the COPPA Rules?," in *2014 Twelfth Annual International Conference on Privacy, Security and Trust*. IEEE, 2014, pp. 1–9.

[235] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *International conference on machine learning*. PMLR, 2021, pp. 12310–12320.

[236] Aaron van den Oord, Yazhe Li, and Oriol Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[237] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer, "Lit: Zero-shot transfer with locked-image text tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18123–18133.

[238] Han Fang, Pengfei Xiong, Luhui Xu, and Yu Chen, "Clip2video: Mastering video-text retrieval via image clip," *arXiv preprint arXiv:2106.11097*, 2021.

[239] Jesús Andrés Portillo-Quintero, José Carlos Ortiz-Bayliss, and Hugo Terashima-Marín, "A straightforward framework for video retrieval using clip," in *Mexican Conference on Pattern Recognition*. Springer, 2021, pp. 3–12.

[240] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.

[241] Daniela Massiceti, Camilla Longden, Agnieszka Slowik, Samuel Wills, Martin Grayson, and Cecily Morrison, "Explaining CLIP's performance disparities on data from blind/low vision users," *arXiv preprint arXiv:2311.17315*, 2023.

[242] Sandhini Agarwal, Gretchen Krueger, Jack Clark, Alec Radford, Jong Wook Kim, and Miles Brundage, "Evaluating clip: towards characterization of broader capabilities and downstream implications," *arXiv preprint arXiv:2108.02818*, 2021.

[243] Sasha Luccioni, Christopher Akiki, Margaret Mitchell, and Yacine Jernite, "Stable bias: Evaluating societal representations in diffusion models," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[244] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas, "Masked siamese networks for label-efficient learning," in *European Conference on Computer Vision*. Springer, 2022, pp. 456–473.

[245] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[246] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier, "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 67–78, 2014.

[247] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[248] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Bin Xiao, Ce Liu, Lu Yuan, and Jianfeng Gao, "Unified contrastive learning in image-text-label space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19163–19173.

[249] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[250] Ilya Loshchilov and Frank Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[251] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[252] Jesse Vig, "A Multiscale Visualization of Attention in the Transformer Model," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Florence, Italy, July 2019, pp. 37–42, Association for Computational Linguistics.

[253] Google Play Console Help, "Manage target audience and app content settings," https://support.google.com/googleplay/android-developer/answer/9867159?hl=en.

[254] D Denipitiyage, B Silva, K Gunathilaka, S Seneviratne, A Mahanti, A Seneviratne, and S Chawla, "Detecting and Characterising Mobile App Metamorphosis in Google Play Store," *arXiv preprint arXiv:2407.14565*, 2024.

[255] "How we fought bad apps and bad actors in 2023," https://security.googleblog.com/2024/04/how-we-fought-bad-apps-and-bad-actors-in-2023.html, 2024, Accessed: 2024-08-07.

[256] Magdalena Steinböck, Jakob Bleier, Mikka Rainer, Tobias Urban, Christine Utz, and Martina Lindorfer, "Comparing Apples to Androids: Discovery, Retrieval, and Matching ofand Android Apps for Cross-Platform Analyses," in *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 2024.

[257] Apple, "Age ratings values and definitions," https://developer.apple.com/help/app-store-connect/reference/age-ratings-values-and-definitions, 2025.

[258] Google, "App Discovery with Google Play, Part 3: Machine Learning to Fight Spam and Abuse at Scale," https://research.google/blog/app-discovery-with-google-play-part-3-machine-learning-to-fight-spam-and-abuse-at-scale/, Mar. 2015.

[259] Google, "Keeping Google Play safe for users and developers: June 29, 2023," https://support.google.com/googleplay/android-developer/answer/13721042?hl=en, 2023.

[260] Hamad Ibrahim, "Google Play review times: Expectations and tips to streamline approval," https://median.co/blog/google-play-review-times-what-to-expect-and-how-to-streamline-approval, Nov. 2024.

[261] Play Store, "Creating apps and games for children and families," https://play.google.com/console/about/programs/families/, 2015.

[262] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn, "Direct preference optimization: Your language model is secretly a reward model," *Advances in neural information processing systems*, vol. 36, pp. 53728–53741, 2023.

[263] K. Thor Jensen, "11 iPhone Apps that Got Banned and Why," https://au.pcmag.com/iphone-apps/95993/11-iphone-apps-that-got-banned-and-why, 2022.

[264] Australasian Legal Information Institute, "ONLINE SAFETY ACT 2021 - SECT 105," https://www.austlii.edu.au/cgi-bin/viewdoc/au/legis/cth/consol_act/osa2021154/s105.html, 2021.

[265] Australia eSafety commissioner, "Illegal and restricted online content," https://www.esafety.gov.au/key-topics/Illegal-restricted-content, 2024.

[266] Australasian Legal Information Institute, "Online content regulation.," https://www.austlii.edu.au/cgi-bin/viewdb/au/legis/cth/consol_act/bsa1992214/, 1992.

[267] Leon Y Xiao and Mie Lange Lund, "Non-compliance with and non-enforcement of UK loot box industry self-regulation on the Apple App Store: a longitudinal study on poor implementation," *Royal Society Open Science*, vol. 12, no. 5, pp. 250704, 2025.

[268] Marcus Carter, Tianyi Zhangshao, Taylor Hardwick, Ben Egliston, and Leon Y Xiao, "Investigating mobile games' compliance with Australia's 2024 mandatory minimum age classifications scheme for gambling-like mechanics," *Available at SSRN*, 2025.

[269] Ke-Li Chiu, Annie Collins, and Rohan Alexander, "Detecting hate speech with gpt-3," *arXiv preprint arXiv:2103.12407*, 2021.

[270] Keyan Guo, Alexander Hu, Jaden Mu, Ziheng Shi, Ziming Zhao, Nishant Vishwamitra, and Hongxin Hu, "An investigation of large language models for real-world hate speech detection," in *2023 International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2023, pp. 1568–1573.

[271] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee, "Hatexplain: A benchmark dataset for explainable hate speech detection," in *Proceedings of the AAAI conference on artificial intelligence*, 2021, vol. 35, pp. 14867–14875.

[272] Han Wang, Rui Yang Tan, and Roy Ka-Wei Lee, "Cross-Modal Transfer from Memes to Videos: Addressing Data Scarcity in Hateful Video Detection," in *Proceedings of the ACM on Web Conference 2025*, 2025, pp. 5255–5263.

[273] Apple Inc., "Choosing a category," https://developer.apple.com/app-store/categories/, 2025.

[274] Apple Newsroom., "Apple expands tools to help parents protect kids and teens online.," https://www.apple.com/au/newsroom/2025/06/apple-expands-tools-to-help-parents-protect-kids-and-teens-online/, 2025.

[275] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al., "Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features," *arXiv preprint arXiv:2502.14786*, 2025.

[276] Lingchen Meng, Jianwei Yang, Rui Tian, Xiyang Dai, Zuxuan Wu, Jianfeng Gao, and Yu-Gang Jiang, "Deepstack: Deeply stacking visual tokens is surprisingly simple and effective for lmms," *Advances in Neural Information Processing Systems*, vol. 37, pp. 23464–23487, 2024.

[277] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee, "Improved Baselines with Visual Instruction Tuning," 2023.

[278] Dan Hendrycks and Kevin Gimpel, "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks," in *International Conference on Learning Representations*, 2017.

[279] Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, Fangyuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun, "Revisiting out-of-distribution robustness in nlp: Benchmarks, analysis, and llms evaluations," *Advances in Neural Information Processing Systems*, vol. 36, pp. 58478–58507, 2023.

[280] Youssef Shoeb, Azarm Nowzad, and Hanno Gottschalk, "Out-of-distribution segmentation in autonomous driving: Problems and state of the art," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 4310–4320.

[281] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017, ICML'17, p. 1321–1330.

[282] Weitang Liu, Xiaoyun Wang, John D. Owens, and Yixuan Li, "Energy-based out-of-distribution detection," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, NIPS '20.

[283] Yiyou Sun et al., "React: Out-of-distribution detection with rectified activations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 144–157, 2021.

[284] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang, "Vim: Out-of-distribution with virtual-logit matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4921–4930.

[285] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich, "Deep Anomaly Detection with Outlier Exposure," in *International Conference on Learning Representations*, 2019.

[286] Qing Yu et al., "Unsupervised out-of-distribution detection by maximum classifier discrepancy," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9518–9526.

[287] Jingkang Yang, Haoqi Wang, Litong Feng, Xiaopeng Yan, Huabin Zheng, Wayne Zhang, and Ziwei Liu, "Semantically coherent out-of-distribution detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8301–8309.

[288] Jingyang Zhang, Nathan Inkawhich, Randolph Linderman, Yiran Chen, and Hai Li, "Mixture outlier exposure: Towards out-of-distribution detection in fine-grained environments," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 5531–5540.

[289] Abhijit Bendale et al., "Towards open set deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1563–1572.

[290] Naveen Karunanayake, Suranga Seneviratne, and Sanjay Chawla, "CRAFT: Class Ranking Aware Fine-Tuning for Enhanced Out-of-Distribution Detection," in *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2025, pp. 4119–4128.

[291] Naveen Karunanayake, Suranga Seneviratne, and Sanjay Chawla, "ExCeL: Combined Extreme and Collective Logit Information for Enhancing Out-of-Distribution Detection," *arXiv preprint arXiv:2311.14754*, 2023.

[292] John I Marden, *Analyzing and modeling rank data*, CRC Press, 1996.

[293] Jingyang Zhang, Jingkang Yang, Pengyun Wang, Haoqi Wang, Yueqian Lin, Haoran Zhang, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, et al., "OpenOOD v1. 5: Enhanced Benchmark for Out-of-Distribution Detection," *arXiv preprint arXiv:2306.09301*, 2023.

[294] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin, "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[295] Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li, "Mitigating neural network overconfidence with logit normalization," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23631–23644.

[296] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song, "Using self-supervised learning can improve model robustness and uncertainty," *Advances in neural information processing systems*, vol. 32, 2019.

[297] Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu, "Extremely Simple Activation Shaping for Out-of-Distribution Detection," in *The Eleventh International Conference on Learning Representations*, 2023.

[298] Yiyou Sun et al., "Dice: Leveraging sparsification for out-of-distribution detection," in *European Conference on Computer Vision*. Springer, 2022, pp. 691–708.

[299] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira, "Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10951–10960.

[300] Yifei Ming, Yiyou Sun, Ousmane Dia, and Yixuan Li, "How to Exploit Hyperspherical Embeddings for Out-of-Distribution Detection?," in *The Eleventh International Conference on Learning Representations*, 2023.

[301] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin, "Csi: Novelty detection via contrastive learning on distributionally shifted instances," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11839–11852, 2020.

[302] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," *"https://www.cs.utoronto.ca/ kriz/learning-features-2009-TR.pdf"*, 2009.

[303] Ya Le and Xuan Yang, "Tiny imagenet visual recognition challenge," *CS 231N*, vol. 7, no. 7, pp. 3, 2015.

[304] Li Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[305] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[306] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3606–3613.

[307] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[308] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman, "Open-Set Recognition: A Good Closed-Set Classifier is All You Need," in *International Conference on Learning Representations*, 2022.

[309] Julian Bitterwolf, Maximilian Müller, and Matthias Hein, "In or out? fixing imagenet out-of-distribution detection evaluation," in *Proceedings of the 40th International Conference on Machine Learning*, 2023, ICML'23.

[310] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie, "The inaturalist species classification and detection dataset,"

in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8769–8778.

[311] Xixi Liu et al., "GEN: Pushing the Limits of Softmax-Based Out-of-Distribution Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 23946–23955.

[312] Terrance DeVries et al., "Learning confidence for out-of-distribution detection in neural networks," *arXiv preprint arXiv:1802.04865*, 2018.

[313] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan, "A simple fix to mahalanobis distance for improving near-ood detection," *arXiv preprint arXiv:2106.09022*, 2021.

[314] Dan Hendrycks, Steven Basart, Mantas Mazeika, Andy Zou, Joseph Kwon, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song, "Scaling Out-of-Distribution Detection for Real-World Settings," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*. 2022, vol. 162 of *Proceedings of Machine Learning Research*, pp. 8759–8773, PMLR.

[315] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li, "Out-of-distribution detection with deep nearest neighbors," in *International Conference on Machine Learning*. PMLR, 2022, pp. 20827–20840.

[316] Xinyi Hou, Yanjie Zhao, and Haoyu Wang, "On the (in) security of llm app stores," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 317–335.

[317] Aakash Sorathiya and Gouri Ginde, "Beyond Keywords: A Context-based Hybrid Approach to Mining Ethical Concern-related App Reviews," *arXiv preprint arXiv:2411.07398*, 2024.

[318] Chuanqi Tao, Hongjing Guo, and Zhiqiu Huang, "Identifying security issues for mobile applications based on user review summarization," *Information and Software Technology*, vol. 122, pp. 106290, 2020.

[319] Yan Zhuang and Yanru Zhang, "Yet at Memotion 2.0 2022: Hate speech detection combining bilstm and fully connected layers," in *Proceedings of de-factify: Workshop on multimodal fact checking and hate speech detection, CEUR*, 2022.

[320] Zhiyu Ma, Shaowen Yao, Liwen Wu, Song Gao, and Yunqi Zhang, "Hateful memes detection based on multi-task learning," *Mathematics*, vol. 10, no. 23, pp. 4525, 2022.

[321] Aneri Rana and Sonali Jha, "Emotion based hate speech detection using multimodal learning," *arXiv preprint arXiv:2202.06218*, 2022.

[322] Claire Wonjeong Jo, Magdalena Wojcieszak, et al., "Harmful YouTube video detection: A taxonomy of online harm and MLLMs as alternative annotators," *arXiv preprint arXiv:2411.05854*, 2024.

[323] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi, "Unified-io: A unified model for vision, language, and multi-modal tasks," *arXiv preprint arXiv:2206.08916*, 2022.

[324] Shiyu Liang, Yixuan Li, and R. Srikant, "Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks," in *International Conference on Learning Representations*, 2018.

[325] Chandramouli Shama Sastry et al., "Detecting out-of-distribution examples with gram matrices," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8491–8501.

[326] Shu Kong et al., "Opengan: Open-set recognition via open data generation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 813–822.

[327] Rui Huang et al., "On the importance of gradients for detecting distributional shifts in the wild," *Advances in Neural Information Processing Systems*, vol. 34, pp. 677–689, 2021.

[328] Yue Song et al., "Rankfeat: Rank-1 feature removal for out-of-distribution detection," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17885–17898, 2022.

[329] Jinsong Zhang, Qiang Fu, Xu Chen, Lun Du, Zelin Li, Gang Wang, Shi Han, Dongmei Zhang, et al., "Out-of-Distribution Detection based on In-Distribution Data Patterns Memorization with Modern Hopfield Energy," in *The Eleventh International Conference on Learning Representations*, 2022.

[330] Guangyao Chen, Peixi Peng, Xiangqian Wang, and Yonghong Tian, "Adversarial reciprocal points learning for open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 8065–8081, 2021.

[331] Rui Huang et al., "Mos: Towards scaling out-of-distribution detection for large semantic space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8710–8719.

Appendix A

Introduction

A.1 ACB Content Rating Definitions

This appendix presents the content rating descriptors defined by the Australian Classification Board (ACB) and their associated impact across each content rating category.

Categories	ACB Content Rating Descriptor	G	PG	M	MA15+	R18+
Online Interactivity	Online interactivity	✓	✓	✓	✓	✓
	Chat	✓	✓	✓	✓	✓
In-game Purchases	In-game purchases	✗	✗	✓	✓	✓
Themes	Mature themes	v. mild	mild	mod.	strong	high
	Blood and gore	✗	✗	mod.	strong	high
	Bullying themes	v. mild	mild	mod.	strong	high
	Crude humour	v. mild	mild	mod.	strong	high
	Fantasy themes	v. mild	mild	mod.	strong	high
	Gambling themes	v. mild	mild	mod.	strong	high
	Mental health themes	v. mild	mild	mod.	strong	✗
	Horror themes	v. mild	mild	mod.	strong	high
	Injury detail	✗	mild	mod.	strong	high
	Science fiction themes	v. mild	mild	mod.	strong	high
	Self-harm	✗	✗	mod.	strong	high
	Simulated gambling	✗	✗	✗	✗	high
	Suicide references	✗	mild	mod.	strong	high
	Suicide scenes	✗	✗	mod.	strong	high
	Supernatural themes	v. mild	mild	mod.	strong	high
Violence	Violence	v. mild	mild	mod.	strong	high
	References to sexual violence	✗	✗	✗	✗	high
	Action violence	✗	mild	mod.	strong	high
	Battle violence	✗	mild	mod.	strong	high
	Comedic violence	v. mild	mild	mod.	strong	high
	Family violence	✗	mild	mod.	strong	high
	Fantasy violence	v. mild	mild	mod.	strong	high
	Hunting violence	✗	mild	mod.	strong	high
	Horror violence	✗	✗	mod.	strong	high
	Science fiction violence	v. mild	mild	mod.	strong	high

Categories	ACB Content Rating Descriptor	G	PG	M	MA15+	R18+
Violence	Sporting violence	v. mild	mild	mod.	strong	high
	Supernatural violence	v. mild	mild	mod.	strong	high
Language	Coarse language	v. mild	mild	mod.	strong	high
Drug Use	Drug references	✗	mild	mod.	strong	high
	Drug use	✗	mild	mod.	strong	high
	Interactive drug use	✗	✗	✗	✗	high
Nudity	Nudity	v. mild	mild	mod.	strong	high
	Nudity related to incentives and rewards	✗	✗	✗	✗	high
Sex	Sexual references	v. mild	mild	mod.	strong	high
	Sex scenes	✗	mild	mod.	strong	high
	Sexualised imagery	v. mild	mild	mod.	strong	high
	sexual activity related incentives & rewards	✗	✗	✗	✗	high

Table A.1: Australian Classification Board (ACB) content descriptors and their permitted severity across age ratings. Entries indicate the maximum allowed intensity (e.g., very mild, mild, strong, high impact) or availability of each descriptor under the corresponding classification.

Appendix B

Detecting and Characterising Mobile App Metamorphosis in Google Play Store

We discuss the ethical considerations of our work in Appendix B.1, and in Appendix B.2, we provide the comparison of search efficiency between our model and Karunanayake et.al [22]. Finally, in Appendix B.3 and Appendix B.5 we describe app distribution of the validation set and hyper-parameter selection for re-purposed apps.

B.1 Ethics

During the crawling stages, we only crawled publicly available app metadata hosted in Google Play Store, and to avoid any disturbances to its operation, we visited Play Store pages at a very slow rate. Beyond that, we did not process any data related to app users, such as app reviews.

B.2 Search Efficiency

Here, we show the comparison of the efficiency of our proposed method with the method introduced by Karunanayake et al. [22]. Both methods rely on generating embeddings, but their subsequent processes differ significantly. The method [22] requires an extensive grid search to identify optimal parameters, where each parameter ranges from 1 to 9. Using our validation set (Section 3.3), this hyper-parameter tuning took approximately 17.25 hours (62,097.613 seconds). However, this step is performed only once. After hyper-parameter tuning, [22] performs a similarity search using optimised parameters. In contrast, our method generates FAISS indices and uses those indices to find the most similar counterpart. We tested this on the test set (Section 3.3) by progressively increasing the size of the queried dataset. Figure B.1 shows the time taken to find a counterpart for a single app across increasing dataset sizes. The time taken by [22] increases exponentially as the queried app space grows. In contrast, our method scales more efficiently, demonstrating near-linear growth. This makes our approach significantly more practical for larger datasets, where [22] becomes infeasible due to the exponential increase in processing time.

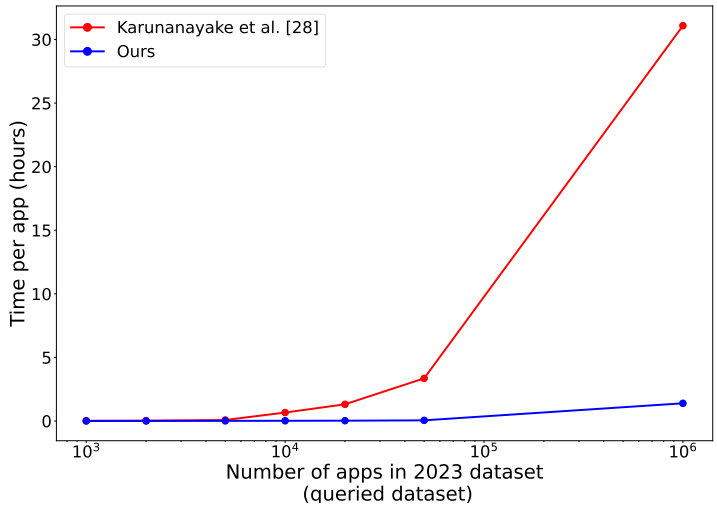


Figure B.1: Comparison of Time Efficiency: Proposed Method vs. Karunanayake et al. [22]

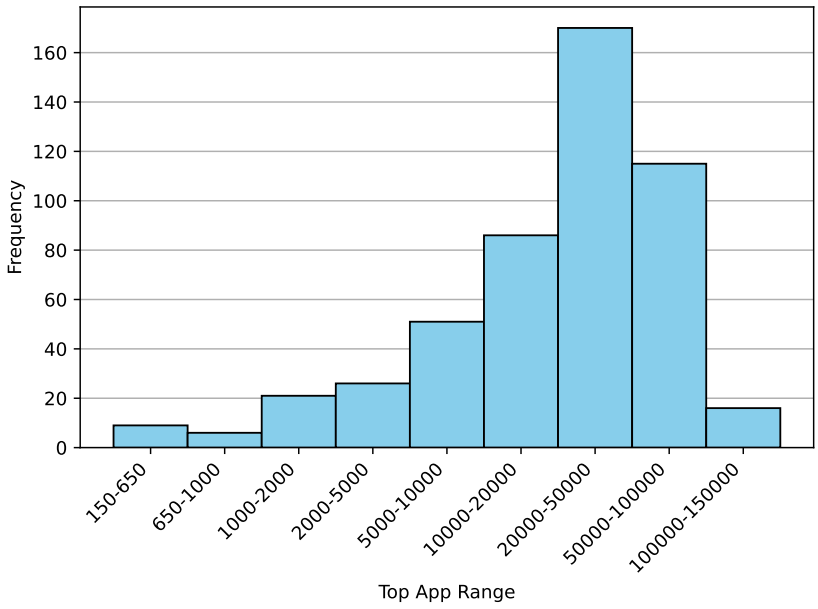


Figure B.2: Distribution of the validation set across dataset

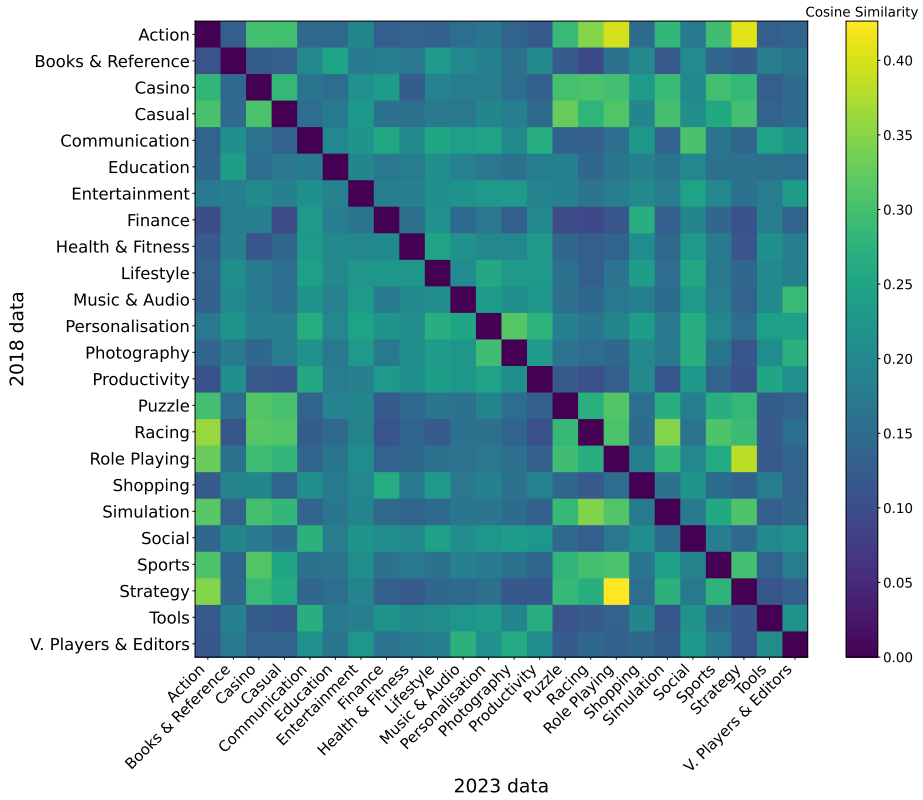


Figure B.3: Average cosine similarity between app genres from two datasets 2018 and 2023.

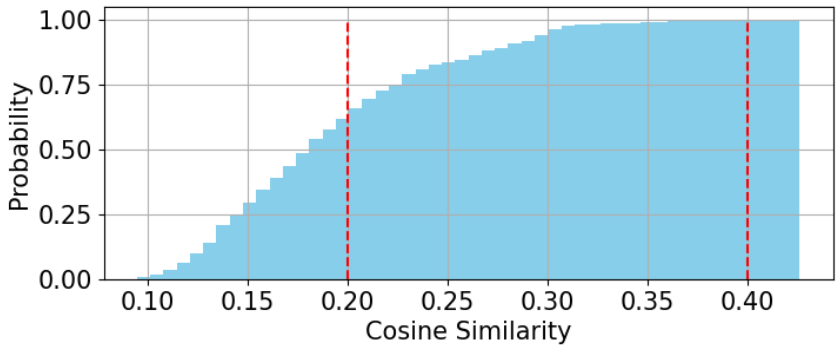


Figure B.4: CDF of average cosine similarity between app genres from two datasets 2018 and 2023.

Table B.1: Harmonic mean on the validation set for different embedding combinations and voting thresholds

	α = 1		α = 2		α = **3**		α = 4		Harmonic mean			
		No		No		No		No			α	
	M.	M.	M.	M.	M.	M.	M.	M.	1	2	3	4
$MPNet_{desc}(M)$	93.0	0.0	-	-	-	-	-	-	0.0	-	-	-
$TFIDF_{name}(T)$	89.6	0.0	-	-	-	-	-	-	0.0	-	-	-
ViT_{icon}	75.6	0.0	-	-	-	-	-	-	0.0	-	-	-
$VGG19^{c}_{icon}$	70.4	0.0	-	-	-	-	-	-	0.0	-	-	-
$StyTr^{2^c}_{icon}$	58.4	0.0	-	-	-	-	-	-	0.0	-	-	-
$VGG19^{s}_{icon}$	74.4	0.0	-	-	-	-	-	-	0.0	-	-	-
$StyTr^{2^s}_{icon}$	62.4	0.0	-	-	-	-	-	-	0.0	-	-	-
$M + T$	88.8	0.0	83.4	56.8	73.6	92.6	-	-	0.0	67.6	82.0	-
$M + ViT_{icon}$	91.8	0.0	70.6	84.0	63.0	95.6	-	-	0.0	76.7	75.9	-
$M + VGG19^{c}_{icon}$	87.6	0.0	68.0	92.6	60.8	96.2	-	-	0.0	78.4	74.5	-
$M + StyTr^{2^c}_{icon}$	89.2	0.0	56.2	96.0	50.2	98.2	-	-	0.0	70.9	66.4	-
$M + ViT_{icon} + T$	90.2	0.0	88.6	45.4	84.8	89.0	56.0	97.6	0.0	60.0	86.8	71.2
$M + VGG19^{c}_{icon} + T$	89.4	0.0	88.6	43.8	84.2	89.0	52.8	98.0	0.0	58.6	86.5	68.6
$M + StyTr^{2^c}_{icon} + T$	88.4	0.0	86.8	45.6	81.8	90.8	43.0	98.8	0.0	59.8	86.1	59.9
$M + T + ViT_{icon} + VGG19^{s}_{icon}$	88.8	0.0	88.4	32.2	86.2	85.6	70.2	95.6	0.0	47.2	85.9	81.0
$M + T + ViT_{icon} + StyTr^{2^s}_{icon}$	90.0	0.0	89.4	35.4	86.2	86.0	65.8	96.4	0.0	50.7	86.1	78.2
$M + T + VGG19^{c}_{icon} + VGG19^{s}_{icon}$	86.2	0.0	85.8	28.8	83.6	86.6	64.4	96.2	0.0	43.1	85.1	77.2
$M + T + VGG19^{c}_{icon} + StyTr^{2^s}_{icon}$	89.2	0.0	88.8	37.6	85.4	88.2	62.6	97.2	0.0	52.8	86.8	76.2
$\mathbf{M + T + StyTr^{2^c}_{icon} + VGG19^{s}_{icon}}$	88.8	0.0	88.4	36.8	86.0	88.2	63.0	97.0	0.0	52.0	**87.1**	76.4
$M + T + StyTr^{2^c}_{icon} + StyTr^{2^s}_{icon}$	89.2	0.0	88.8	37.6	85.4	88.2	62.6	97.2	0.0	37.1	84.5	69.7

B.3 Validation App Distribution

As shown in the Figure B.2, app selection in the validation set ranges from top-150 to top-150k allowing to select a generic value for the hyper-parameter α.

B.4 Modality Selection

To evaluate the effectiveness of each modality, we conducted experiments on the validation set described in Section 3.3. These experiments involved varying the number of modalities used for similarity matching and calculating the harmonic mean of accuracy for both matches and non-matches across four different occurrence count thresholds (α values). Starting with a single modality, we incrementally increased the number of modalities up to five. The findings are depicted in the Table B.1. The results demonstrate that the selected combination of four modalities: MPNet app description embeddings ($MPNET_{desc}$ or M), TF-IDF app name and developer name embeddings ($TFIDF_{name}$ or T), VGG19 style embedding ($VGG19^{s}_{icon}$) and $StyTr^2$ content embeddings ($StyTr^{2c}_{icon}$) for app icon achieves the highest performance. This analysis supports our claim that the chosen combination is the most effective for identifying matches and no-matches.

B.5 Ablation Study - Re-purposed Apps

Having core functionality changes in re-purposed apps, we highlighted that the app description embeddings should deviate significantly from one another. To validate the choice of the 0.2–0.4 similarity threshold, we conducted an experiment comparing the cosine similarity of app descriptions across genres in the 2018 and 2023 datasets with the intuition that dis-similar genres are likely to contain dis-similar descriptions explaining the functionality. In this experiment, for each genre, 100 apps were sampled based on availability, and their average cosine similarity of app description embeddings were computed and portrayed in Figure B.3 and Figure B.4. The average inter-genre cosine similarity is 0.193 and the maximum cosine similarity is 0.43. Furthermore, we observed apps with descriptions in non-English languages producing similarities around 0.15, contributing to noise. To mitigate this and focus on apps that demonstrate significant but interpretable functional shifts, we selected a threshold of 0.2–0.4. This range effectively excludes noisy data.

Appendix C

RankOOD: Class Ranking-based Out-of-Distribution Detection

In this supplementary material, we present additional details and results that were excluded from the main content due to space limitations.

C.1 Comparison with baselines

In the main text (Tab. 2 and Tab. 3), we report results only for baselines that placed in the top five in at least one setting. For completeness, we provide the full results for all 35 baselines in Tab. C.2 (near-OOD) and Tab. C.3 (far-OOD), with key observations summarized in the respective captions.

C.2 Detailed per-dataset OOD detection results

In Section 6.5.1 , we reported average AUROC and FPR95 across all OOD datasets for each ID dataset (Tab. 2 and Tab. 3). For completeness, we provide the per-dataset results for each OOD benchmark. TinyImageNet results in Tab. C.4 and Tab. C.5; CIFAR-100 results in Tab. C.6 and Tab. C.7; and CIFAR-10 results are shown in Tab. C.8 and Tab. C.9.

Table C.1: Per-epoch GPU time (sec.) and per-class ILP runtime (sec.) across different datasets (C-CIFAR, IN-ImageNet).

Method	**C-10**	**C-100**	**IN-200**	**IN-1k**	**Epochs**
LogitNorm	8.33	5.32	1045.50	6614.0	~ 200
CRAFT	17.45	25.44	1354.00	-	~ 100
RankOOD	13.80	14.34	1402.12	6855.7	$\sim 300 - 500$
ILP Runtime	0.0058	0.1718	2.4933	1413.6	-
ILP - Greedy	0.0006	0.0023	0.0177	0.2147	-

Table C.2: Performance comparison in *near-OOD detection*. For each column, the top five methods are marked in **bold**. Note that N/A indicates missing results in OpenOOD. *OE obtains the lowest average FPR95 (34.29), while RankOOD ranks second with 44.79. However, OE's performance is known to be biased toward seen outliers used during training, making its evaluation less reliable [290]. Among methods that do not rely on outliers, RankOOD achieves the best performance for both AUROC and FPR95. Notably, RankOOD achieves state-of-the-art performance on TinyImageNet near-OOD, reducing FPR95 by 4.3% relative to the strongest baseline OE.*

Method	CIFAR-10		CIFAR-100		ImageNet-200		Average	
	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓
Post-hoc inference methods								
OpenMax [289]	87.62 ± 0.29	43.62 ± 2.27	76.41 ± 0.25	56.58 ± 0.73	80.27 ± 0.10	63.48 ± 0.25	81.43	54.56
MSP [278]	88.03 ± 0.25	48.17 ± 3.92	80.27 ± 0.11	**54.80 ± 0.33**	83.34 ± 0.06	54.82 ± 0.35	83.88	52.60
TempScale [281]	88.09 ± 0.31	50.96 ± 4.32	80.90 ± 0.07	**54.49 ± 0.48**	**83.69 ± 0.04**	54.82 ± 0.23	84.23	53.42
ODIN [324]	82.87 ± 1.85	76.19 ± 6.08	79.90 ± 0.11	57.91 ± 0.51	80.27 ± 0.08	66.76 ± 0.26	81.01	66.95
MDS [294]	84.20 ± 2.40	49.90 ± 3.98	58.69 ± 0.09	83.53 ± 0.60	61.93 ± 0.51	79.11 ± 0.31	68.27	70.85
MDSEns [294]	60.43 ± 0.26	92.26 ± 0.20	46.31 ± 0.24	95.88 ± 0.04	54.32 ± 0.24	91.75 ± 0.10	53.69	93.30
RMDS [313]	89.80 ± 0.28	38.89 ± 2.39	80.15 ± 0.11	55.46 ± 0.41	82.57 ± 0.25	**54.02 ± 0.58**	84.17	**49.46**
Gram [325]	58.66 ± 4.83	90.87 ± 1.91	51.66 ± 0.77	92.28 ± 0.29	67.67 ± 1.07	86.40 ± 1.21	59.33	89.85
EBO [282]	87.58 ± 0.46	61.34 ± 4.63	**80.91 ± 0.08**	55.62 ± 0.61	82.50 ± 0.05	60.24 ± 0.57	83.66	59.07
OpenGAN [326]	53.71 ± 7.68	94.48 ± 4.01	65.98 ± 1.26	76.52 ± 2.59	59.79 ± 3.39	84.15 ± 3.85	59.83	85.05
GradNorm [327]	54.90 ± 0.98	94.72 ± 0.82	70.13 ± 0.47	85.58 ± 0.46	72.75 ± 0.48	82.67 ± 0.30	65.93	87.66
ReAct [283]	87.11 ± 0.61	63.56 ± 7.33	80.77 ± 0.05	56.39 ± 0.34	81.87 ± 0.98	62.49 ± 2.19	83.25	60.81
MLS [314]	87.52 ± 0.47	61.32 ± 4.62	**81.05 ± 0.07**	55.47 ± 0.66	82.90 ± 0.04	59.76 ± 0.59	83.82	58.85
KLM [314]	79.19 ± 0.80	87.86 ± 6.37	76.56 ± 0.25	77.92 ± 1.31	80.76 ± 0.08	70.26 ± 0.64	78.84	78.68
VIM [284]	88.68 ± 0.28	44.84 ± 2.31	74.98 ± 0.13	62.63 ± 0.27	78.68 ± 0.24	59.19 ± 0.71	80.78	55.55
KNN [315]	90.64 ± 0.20	34.01 ± 0.38	80.18 ± 0.15	61.22 ± 0.14	81.57 ± 0.17	60.18 ± 0.52	84.13	51.80
DICE [298]	78.34 ± 0.79	70.04 ± 7.64	79.38 ± 0.23	57.95 ± 0.53	81.78 ± 0.14	61.88 ± 0.67	79.83	63.29
RankFeat [328]	79.46 ± 2.52	60.88 ± 4.60	61.88 ± 1.28	80.59 ± 1.10	56.92 ± 1.59	92.06 ± 0.23	66.09	77.84
ASH [297]	75.27 ± 1.04	86.78 ± 1.82	78.20 ± 0.15	65.71 ± 0.24	82.38 ± 0.19	64.89 ± 0.90	78.62	72.46
SHE [329]	81.54 ± 0.51	79.65 ± 3.47	78.95 ± 0.18	59.07 ± 0.25	80.18 ± 0.25	66.80 ± 0.74	80.22	68.51
GEN [311]	88.20 ± 0.30	53.67 ± 3.14	**81.31 ± 0.08**	**54.42 ± 0.33**	**83.68 ± 0.06**	55.20 ± 0.20	**84.40**	54.43
ExCeL [291]	86.89 ± 0.23	66.55 ± 0.43	80.70 ± 0.06	55.21 ± 0.56	82.40 ± 0.04	57.90 ± 0.40	83.33	59.89
Training methods without outliers								
RankOOD (ours)	90.21 ± 0.41	**31.72 ± 0.67**	80.67 ± 0.40	**52.59 ± 0.75**	**85.30 ± 0.18**	**50.05 ± 0.16**	**85.39**	**44.79**
CRAFT[290]	**91.11 ± 0.04**	31.94 ± 1.41	80.90 ± 0.33	53.73 ± 0.62	**83.65 ± 0.41**	**54.62 ± 0.57**	**85.22**	46.76
ConfBranch [312]	89.84 ± 0.24	**31.28 ± 0.66**	71.60 ± 0.62	70.21 ± 0.83	79.10 ± 0.24	61.44 ± 0.34	80.18	54.31
G-ODIN [299]	89.12 ± 0.57	45.54 ± 2.52	77.15 ± 0.28	67.58 ± 0.98	77.28 ± 0.10	69.87 ± 0.46	81.18	61.00
CSI [301]	89.51 ± 0.19	33.66 ± 0.64	71.45 ± 0.27	70.26 ± 0.56	N/A	N/A	80.48	51.96
ARPL [330]	87.44 ± 0.15	40.33 ± 0.70	74.94 ± 0.93	61.56 ± 1.81	82.02 ± 0.10	55.74 ± 0.70	81.47	52.54
MOS [331]	71.45 ± 3.09	78.72 ± 5.86	80.40 ± 0.18	56.05 ± 1.01	69.84 ± 0.46	71.60 ± 0.48	73.90	68.79
LogitNorm [295]	**92.33 ± 0.08**	**29.34 ± 0.81**	78.47 ± 0.31	62.89 ± 0.57	82.66 ± 0.15	56.46 ± 0.37	**84.49**	**49.56**
CIDER [300]	**90.71 ± 0.16**	32.11 ± 0.94	73.10 ± 0.39	72.02 ± 0.31	80.58 ± 1.75	60.10 ± 0.73	81.46	54.74
Training methods with outliers								
OE [285]	**94.82 ± 0.21**	**19.84 ± 0.95**	**88.30 ± 0.10**	**30.73 ± 0.11**	**84.84 ± 0.16**	**52.30 ± 0.67**	**89.32**	**34.29**
MCD [286]	**91.03 ± 0.12**	**30.17 ± 0.06**	77.07 ± 0.32	55.88 ± 0.85	83.62 ± 0.09	**54.71 ± 0.83**	83.91	**46.92**
UDG [287]	89.91 ± 0.25	35.34 ± 0.95	78.02 ± 0.10	61.42 ± 0.48	74.30 ± 1.63	68.89 ± 1.72	80.74	55.22
MixOE [288]	88.73 ± 0.82	51.45 ± 7.78	**80.95 ± 0.20**	55.22 ± 0.49	82.62 ± 0.03	57.97 ± 0.40	84.10	54.88

C.3 Effectiveness of RankOOD-T

Fig. C.1 shows standard logit-based OOD scores (MSP, MaxLogit, EBO (Energy based), and GEN) under RankOOD-T. It can be seen that RankOOD-T performs better across all scoring methods, including RankOOD-S defined in Eq. 6, compared to cross-entropy (CE) training. *This indicates that the observed performance gains primarily stem from the RankOOD-T objective rather than the RankOOD-S function itself.* In particular, ListMLE training improve s AUROC by at least 1.5% and reduces FPR95 by over 7% compared to CE training. Nonetheless, as shown in Fig C.1, logit-based OOD scores can be used in open-world settings to obtain on-par performance after RankOOD-T.

Table C.3: Performance comparison in *far-OOD detection*. For each column, the top five methods are marked in **bold**. Note that N/A indicates missing results in OpenOOD. *RankOOD ranks third in both AUROC (89.65) and FPR95 (32.04) on far-OOD detection. LogitNorm and G-ODIN achieve the strongest results, outperforming RankOOD by roughly 2.7% in FPR95. Nonetheless, RankOOD surpasses G-ODIN on CIFAR-10 and TinyImageNet and exceeds LogitNorm on CIFAR-100. Moreover, RankOOD outperforms OE on CIFAR-100 and ImageNet-200, while achieving an overall AUROC within 1% of the best-performing outlier based methods.*

Method	CIFAR-10		CIFAR-100		TinyImageNet		Average	
	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓	AUROC ↑	FPR95 ↓
Post-hoc inference methods								
OpenMax [289]	89.62 ± 0.19	29.69 ± 1.21	79.48 ± 0.41	54.50 ± 0.68	90.20 ± 0.17	33.12 ± 0.66	86.43	39.10
MSP [278]	90.73 ± 0.43	31.72 ± 1.84	77.76 ± 0.44	58.70 ± 1.06	90.13 ± 0.09	35.43 ± 0.38	86.21	41.95
TempScale [281]	90.97 ± 0.52	33.48 ± 2.39	78.74 ± 0.51	57.94 ± 1.14	90.82 ± 0.09	34.00 ± 0.37	86.84	41.81
ODIN [324]	87.96 ± 0.61	57.62 ± 4.24	79.28 ± 0.21	58.86 ± 0.79	91.71 ± 0.19	34.23 ± 1.05	86.32	50.24
MDS [294]	89.72 ± 1.36	32.22 ± 3.40	69.39 ± 1.39	72.26 ± 1.56	74.72 ± 0.26	61.66 ± 0.27	77.94	55.38
MDSEns [294]	73.90 ± 0.27	61.47 ± 0.48	66.00 ± 0.69	66.74 ± 1.04	69.27 ± 0.57	80.96 ± 0.38	69.72	69.72
RMDS [313]	92.20 ± 0.21	25.35 ± 0.73	**82.92 ± 0.42**	52.81 ± 0.63	88.06 ± 0.34	32.45 ± 0.79	87.73	36.87
Gram [313]	71.73 ± 3.20	72.34 ± 6.73	73.36 ± 1.08	64.44 ± 2.37	71.19 ± 0.24	84.36 ± 0.78	72.09	73.71
EBO [282]	91.21 ± 0.92	41.69 ± 5.32	79.77 ± 0.61	56.59 ± 1.38	90.86 ± 0.21	34.86 ± 1.30	87.28	44.38
OpenGAN [326]	54.61 ± 15.5	83.52 ± 11.63	67.88 ± 7.16	70.49 ± 7.38	73.15 ± 4.07	64.16 ± 9.33	65.21	72.72
GradNorm [327]	57.55 ± 3.22	91.90 ± 2.23	69.14 ± 1.05	83.68 ± 1.92	84.26 ± 0.87	66.45 ± 0.22	70.32	80.68
ReAct [283]	90.42 ± 1.41	44.90 ± 8.37	80.39 ± 0.49	54.20 ± 1.56	**92.31 ± 0.56**	28.50 ± 0.95	87.71	42.53
MLS [314]	91.10 ± 0.89	41.68 ± 5.27	79.67 ± 0.57	56.73 ± 1.33	91.11 ± 0.19	34.03 ± 1.21	87.29	44.15
KLM [314]	82.68 ± 0.21	78.31 ± 4.84	76.24 ± 0.52	71.65 ± 2.01	88.53 ± 0.11	40.90 ± 1.08	82.48	63.62
VIM [284]	93.48 ± 0.24	25.05 ± 0.52	81.70 ± 0.62	**50.74 ± 1.00**	91.26 ± 0.19	**27.20 ± 0.30**	**88.81**	**34.33**
KNN [315]	92.96 ± 0.14	24.27 ± 0.40	**82.40 ± 0.17**	53.65 ± 0.28	**93.16 ± 0.22**	**27.27 ± 0.75**	**89.51**	35.06
DICE [298]	84.23 ± 1.89	51.76 ± 4.42	80.01 ± 0.18	56.25 ± 0.60	90.80 ± 0.31	36.51 ± 1.18	85.01	48.17
RankFeat [328]	75.87 ± 5.06	57.44 ± 7.99	67.10 ± 1.42	69.45 ± 1.01	38.22 ± 3.85	97.72 ± 0.75	60.40	74.87
ASH [297]	78.49 ± 2.58	79.03 ± 4.22	80.58 ± 0.66	59.20 ± 2.46	**93.90 ± 0.27**	**27.29 ± 1.12**	84.32	55.17
SHE [329]	85.32 ± 1.43	66.48 ± 5.98	76.92 ± 1.16	64.12 ± 2.70	89.81 ± 0.61	42.17 ± 1.24	84.02	57.59
GEN [311]	91.35 ± 0.69	34.73 ± 1.58	79.68 ± 0.75	56.71 ± 1.59	91.36 ± 0.10	32.10 ± 0.59	87.46	41.18
ExCeL [291]	91.69 ± 0.18	40.03 ± 0.84	**82.04 ± 0.90**	**52.24 ± 1.90**	91.97 ± 0.27	28.45 ± 0.80	88.57	40.24
Training methods without outliers								
RankOOD (ours)	93.19 ± 0.84	20.96 ± 2.55	**83.63 ± 1.06**	**47.44 ± 0.80**	92.14 ± 0.20	**27.73 ± 0.33**	**89.65**	**32.04**
CRAFT [290]	93.94 ± 0.20	**19.40 ± 0.88**	82.03 ± 0.34	**51.86 ± 0.49**	90.88 ± 0.89	32.67 ± 1.13	**88.95**	34.64
ConfBranch [312]	92.85 ± 0.29	21.48 ± 0.94	68.90 ± 1.83	71.82 ± 3.39	90.43 ± 0.18	34.75 ± 0.63	84.06	42.68
G-ODIN [299]	**95.51 ± 0.31**	21.45 ± 1.91	**85.67 ± 1.58**	**42.68 ± 3.19**	**92.33 ± 0.11**	30.18 ± 0.49	**91.17**	**31.44**
CSI [301]	92.00 ± 0.30	26.42 ± 0.29	66.31 ± 1.21	76.92 ± 1.29	N/A	N/A	79.16	51.67
ARPL [330]	89.31 ± 0.32	32.39 ± 0.74	73.69 ± 1.80	63.14 ± 2.53	89.23 ± 0.11	36.46 ± 0.08	84.08	44.00
MOS [331]	76.41 ± 5.93	62.90 ± 6.62	80.17 ± 1.21	57.28 ± 3.29	80.46 ± 0.92	51.56 ± 0.42	79.01	57.25
LogitNorm [295]	**96.74 ± 0.06**	**13.81 ± 0.20**	81.53 ± 1.26	53.61 ± 3.45	**93.04 ± 0.21**	**26.11 ± 0.52**	**90.44**	**31.18**
CIDER [300]	**94.71 ± 0.36**	**20.72 ± 0.85**	80.49 ± 0.68	54.22 ± 1.24	90.66 ± 1.68	30.17 ± 2.75	88.62	35.04
Training methods with outliers								
OE [285]	**96.00 ± 0.13**	**13.13 ± 0.53**	81.41 ± 1.49	54.82 ± 2.79	89.02 ± 0.18	34.17 ± 0.56	88.81	**34.04**
MCD [286]	91.00 ± 1.10	32.03 ± 4.21	74.72 ± 0.78	54.39 ± 1.34	88.94 ± 0.10	29.93 ± 0.30	84.89	38.78
UDG [287]	**94.06 ± 0.90**	**20.35 ± 2.41**	79.59 ± 1.77	59.00 ± 3.35	82.09 ± 2.78	62.04 ± 5.99	85.25	47.13
MixOE [288]	91.93 ± 0.69	33.84 ± 4.77	76.40 ± 1.44	63.88 ± 2.48	88.27 ± 0.41	40.93 ± 0.29	85.53	46.22

C.4 Scalability and Computational Cost

GPU Time: Table C.1 reports per-epoch GPU training time (seconds). As the number of classes increases, all methods exhibit comparable training costs. RankOOD incurs approximately 34% higher GPU time than LogitNorm in TinyImageNet due to the ListMLE objective, which requires computing probabilities over full permutations. *We clarify that RankOOD-T follows the same pre-training paradigm as LogitNorm with a total of 300/500 epochs and doesn't need extra fine-tuning epochs like CRAFT as canonical classes can be derived from a SOTA pre-trained model.*

ILP Runtime: Tab. C.1 reports ILP rank execution time. ILP time grows exponentially with the number of ranks and classes, with worst-case complexity $O(2^{C\times K})$ where C-classes and K-ranks. However, this can be addressed using a greedy approach with complexity $O(CK.log(K))$, significantly reducing computational cost.

Table C.4: FPR95 (% ↓) of various methods for different OOD datasets when TinyImageNet is ID. For each column, the top five methods are marked in **bold**. Note that N/A indicates that results are not reported in OpenOOD. *RankOOD achieves SOTA performance in near-OOD and ranks within the top three methods in two out of three far-OOD datasets.*

Method	Near OOD			Far OOD			
	SSB-hard	NINCO	Average	iNaturalist	Textures	OpenImage-O	Average
Post-hoc inference methods							
OpenMax [289]	72.37 ± 0.11	54.59 ± 0.54	63.48 ± 0.25	24.53 ± 0.96	36.80 ± 0.55	38.03 ± 0.49	33.12 ± 0.66
MSP [278]	66.00 ± 0.10	43.65 ± 0.75	54.82 ± 0.35	26.48 ± 0.73	44.58 ± 0.68	35.23 ± 0.18	35.43 ± 0.38
TempScale [281]	66.43 ± 0.26	**43.21 ± 0.70**	54.82 ± 0.23	24.39 ± 0.79	43.57 ± 0.77	34.04 ± 0.31	34.00 ± 0.37
ODIN [324]	73.51 ± 0.38	60.00 ± 0.80	66.76 ± 0.26	22.39 ± 1.87	42.99 ± 1.56	37.30 ± 0.59	34.23 ± 1.05
MDS [294]	83.65 ± 0.47	74.57 ± 0.15	79.11 ± 0.31	58.53 ± 0.75	58.16 ± 0.84	68.29 ± 0.28	61.66 ± 0.27
MDSEns [294]	92.13 ± 0.05	91.36 ± 0.16	91.75 ± 0.10	83.37 ± 0.70	72.27 ± 0.48	87.26 ± 0.10	80.96 ± 0.38
RMDS [313]	**65.91 ± 0.27**	**42.13 ± 1.04**	**54.02 ± 0.58**	24.70 ± 0.90	37.80 ± 1.32	34.85 ± 0.31	32.45 ± 0.79
Gram [325]	85.68 ± 0.85	87.13 ± 1.89	86.40 ± 1.21	85.54 ± 0.40	80.87 ± 1.20	86.66 ± 1.27	84.36 ± 0.78
EBO [282]	69.77 ± 0.32	50.70 ± 0.89	60.24 ± 0.57	26.41 ± 2.29	41.43 ± 1.85	36.74 ± 1.14	34.86 ± 1.30
OpenGAN [326]	88.07 ± 2.23	80.23 ± 5.71	84.15 ± 3.85	60.13 ± 9.79	66.00 ± 9.97	66.34 ± 8.44	64.16 ± 9.33
GradNorm [327]	82.17 ± 0.62	83.17 ± 0.21	82.67 ± 0.30	61.31 ± 2.86	66.88 ± 3.59	71.16 ± 0.23	66.45 ± 0.22
ReAct [283]	71.51 ± 1.92	53.47 ± 2.46	62.49 ± 2.19	22.97 ± 2.25	**29.67 ± 1.35**	32.86 ± 0.74	28.50 ± 0.95
MLS [314]	69.64 ± 0.37	49.87 ± 0.94	59.76 ± 0.59	25.09 ± 2.04	41.25 ± 1.86	35.76 ± 0.74	34.03 ± 1.21
KLM [314]	78.19 ± 2.30	62.33 ± 2.66	70.26 ± 0.64	26.66 ± 1.61	50.24 ± 1.26	45.81 ± 0.59	40.90 ± 1.08
VIM [284]	71.28 ± 0.49	47.10 ± 1.10	59.19 ± 0.71	27.34 ± 0.38	**20.39 ± 0.17**	33.86 ± 0.63	**27.20 ± 0.30**
KNN [315]	73.71 ± 0.31	46.64 ± 0.73	60.18 ± 0.52	24.46 ± 1.06	**24.45 ± 0.29**	32.90 ± 1.12	**27.27 ± 0.75**
DICE [298]	70.84 ± 0.30	52.91 ± 1.20	61.88 ± 0.67	29.66 ± 2.62	40.96 ± 1.87	38.91 ± 1.16	36.51 ± 1.18
RankFeat [328]	90.79 ± 0.37	93.32 ± 0.11	92.06 ± 0.23	98.00 ± 0.80	99.40 ± 0.68	95.77 ± 0.85	97.72 ± 0.75
ASH [297]	72.14 ± 0.97	57.63 ± 0.98	64.89 ± 0.90	22.49 ± 2.24	**25.65 ± 0.80**	33.72 ± 0.97	**27.29 ± 1.12**
SHE [329]	72.64 ± 0.30	60.96 ± 1.33	66.80 ± 0.74	34.38 ± 3.48	45.58 ± 2.42	46.54 ± 1.34	42.17 ± 1.24
GEN [311]	66.79 ± 0.26	43.61 ± 0.61	55.20 ± 0.20	**22.03 ± 0.98**	42.01 ± 0.92	**32.25 ± 0.31**	32.10 ± 0.59
ExCeL [291]	69.28 ± 0.60	46.51 ± 0.20	57.90 ± 0.40	**22.29 ± 1.00**	30.14 ± 0.64	32.91 ± 0.76	28.45 ± 0.80
Training methods without outliers							
RankOOD (Ours)	**60.68 ± 0.61**	**39.43 ± 0.99**	**50.05 ± 0.16**	**19.42 ± 0.71**	33.81 ± 0.95	**29.97 ± 0.56**	**27.73 ± 0.33**
CRAFT [290]	69.07 ± 0.39	**40.40 ± 1.13**	**54.62 ± 0.57**	24.89 ± 1.33	38.73 ± 4.55	33.48 ± 1.74	32.67 ± 1.13
ConfBranch [312]	72.24 ± 0.37	50.63 ± 0.60	61.44 ± 0.34	23.84 ± 0.40	42.42 ± 2.27	37.99 ± 0.09	34.75 ± 0.63
G-ODIN [299]	78.23 ± 0.70	61.52 ± 0.64	69.87 ± 0.46	26.13 ± 0.77	**28.98 ± 1.15**	35.43 ± 0.43	30.18 ± 0.49
CSI [301]	N/A	N/A	N/A	N/A	N/A	N/A	N/A
ARPL [330]	**65.73 ± 0.51**	45.75 ± 0.89	55.74 ± 0.70	29.32 ± 0.64	42.87 ± 1.09	37.20 ± 0.69	36.46 ± 0.08
MOS [331]	74.35 ± 0.32	68.85 ± 0.68	71.60 ± 0.48	49.55 ± 0.73	51.27 ± 1.02	53.86 ± 0.30	51.56 ± 0.42
LogitNorm [295]	67.46 ± 0.21	45.46 ± 0.69	56.46 ± 0.37	**15.70 ± 0.61**	32.13 ± 0.67	**30.49 ± 0.62**	**26.11 ± 0.52**
CIDER [300]	75.50 ± 0.68	44.69 ± 0.88	60.10 ± 0.73	26.54 ± 2.27	31.51 ± 3.68	**32.47 ± 2.40**	30.17 ± 2.75
Training methods with outliers							
OE [285]	**64.67 ± 0.25**	**39.93 ± 1.13**	**52.30 ± 0.67**	27.03 ± 0.47	41.92 ± 1.69	33.56 ± 0.46	34.17 ± 0.56
MCD [286]	**65.69 ± 0.36**	43.74 ± 1.32	**54.71 ± 0.83**	**21.74 ± 0.31**	38.11 ± 0.93	**29.93 ± 0.19**	29.93 ± 0.30
UDG [287]	75.84 ± 1.86	61.94 ± 1.61	68.89 ± 1.72	49.26 ± 7.88	71.94 ± 5.25	64.92 ± 5.07	62.04 ± 5.99
MixOE [288]	68.26 ± 0.48	47.69 ± 0.95	57.97 ± 0.40	30.84 ± 0.45	51.44 ± 0.61	40.51 ± 1.06	40.93 ± 0.29

C.5 RankOOD Score Example

We provide a step-by-step example of RankOOD-S computation in Figure C.2.

C.6 Conditional Probability Matrices

In Section 6.5.3 , we reported class-conditional probability (CP) matrices for only five CIFAR-10 classes. Here, we provide the complete CP matrices for all ten classes under both CE and RankOOD training. For RankOOD, we additionally report CP matrices for all OOD datasets as well as the ID dataset (CIFAR-10). As shown in Fig. C.3, each matrix reflects the model's ability to preserve the class-wise canonical rank order. Under conventional CE training, a sample predicted as Airplane achieves a probability of 0.29 for correctly identifying the rank-4 label, conditioned on all preceding ranks (rank-1 through rank-3) being accurately predicted. In contrast, RankOOD training substantially improves the model's ability to maintain deeper ranking consistency on in-distribution (ID) data: for samples classified as Airplane,

Table C.5: AUROC (% ↑) of various methods for different OOD datasets when TinyImageNet is ID. For each column, the top five methods are marked in **bold**. Note that N/A indicates that results are not reported in OpenOOD. *RankOOD ranks within the top three methods in two out of three far-OOD datasets while achieving SOTA performance in near-OOD setting.*

Method	Near OOD			Far OOD			
	SSB-hard	NINCO	Average	iNaturalist	Textures	OpenImage-O	Average
Post-hoc inference methods							
OpenMax [289]	77.53 ± 0.08	83.01 ± 0.17	80.27 ± 0.10	92.32 ± 0.32	90.21 ± 0.07	88.07 ± 0.14	90.20 ± 0.17
MSP [278]	80.38 ± 0.03	86.29 ± 0.11	83.34 ± 0.06	92.80 ± 0.25	88.36 ± 0.13	89.24 ± 0.02	90.13 ± 0.09
TempScale [281]	**80.71 ± 0.02**	**86.67 ± 0.08**	**83.69 ± 0.04**	93.39 ± 0.25	89.24 ± 0.11	89.84 ± 0.02	90.82 ± 0.09
ODIN [324]	77.19 ± 0.06	83.34 ± 0.12	80.27 ± 0.08	**94.37 ± 0.41**	90.65 ± 0.20	90.11 ± 0.15	91.71 ± 0.19
MDS [294]	58.38 ± 0.58	65.48 ± 0.46	61.93 ± 0.51	75.03 ± 0.76	79.25 ± 0.33	69.87 ± 0.14	74.72 ± 0.26
MDSEns [294]	50.46 ± 0.36	58.18 ± 0.42	54.32 ± 0.24	62.16 ± 0.73	80.70 ± 0.48	64.96 ± 0.51	69.27 ± 0.57
RMDS [313]	80.20 ± 0.23	84.94 ± 0.28	82.57 ± 0.25	90.64 ± 0.46	86.77 ± 0.38	86.77 ± 0.22	88.06 ± 0.34
Gram [325]	65.95 ± 1.08	69.40 ± 1.07	67.67 ± 1.07	65.30 ± 0.20	80.53 ± 0.37	67.72 ± 0.58	71.19 ± 0.24
EBO [282]	79.83 ± 0.02	85.17 ± 0.11	82.50 ± 0.05	92.55 ± 0.50	90.79 ± 0.16	89.23 ± 0.26	90.86 ± 0.21
OpenGAN [326]	55.08 ± 1.84	64.49 ± 4.98	59.79 ± 3.39	75.32 ± 3.32	70.58 ± 4.66	73.54 ± 4.48	73.15 ± 4.07
GradNorm [327]	72.12 ± 0.43	73.39 ± 0.63	72.75 ± 0.48	86.06 ± 1.90	86.07 ± 0.36	80.66 ± 1.09	84.26 ± 0.87
ReAct [283]	78.97 ± 1.33	84.76 ± 0.64	81.87 ± 0.98	93.65 ± 0.88	**92.86 ± 0.47**	**90.40 ± 0.35**	**92.31 ± 0.56**
MLS [314]	80.15 ± 0.01	85.65 ± 0.09	82.90 ± 0.04	93.12 ± 0.45	90.60 ± 0.16	89.62 ± 0.21	91.11 ± 0.19
KLM [314]	77.56 ± 0.18	83.96 ± 0.12	80.76 ± 0.08	91.80 ± 0.21	86.13 ± 0.12	87.66 ± 0.17	88.53 ± 0.11
VIM [284]	74.04 ± 0.31	83.32 ± 0.19	78.68 ± 0.24	90.96 ± 0.36	**94.61 ± 0.12**	88.20 ± 0.18	91.26 ± 0.19
KNN [315]	77.03 ± 0.23	86.10 ± 0.12	81.57 ± 0.17	**93.99 ± 0.36**	**95.29 ± 0.02**	**90.19 ± 0.32**	**93.16 ± 0.22**
DICE [298]	79.06 ± 0.05	84.49 ± 0.24	81.78 ± 0.14	91.81 ± 0.79	91.53 ± 0.21	89.06 ± 0.34	90.80 ± 0.31
RankFeat [328]	58.74 ± 0.94	55.10 ± 2.52	56.92 ± 1.59	33.08 ± 4.68	29.10 ± 2.57	52.48 ± 4.44	38.22 ± 3.85
ASH [297]	79.52 ± 0.37	85.24 ± 0.08	82.38 ± 0.19	**95.10 ± 0.47**	**94.77 ± 0.19**	**91.82 ± 0.25**	**93.90 ± 0.27**
SHE [329]	78.30 ± 0.20	82.07 ± 0.33	80.18 ± 0.25	91.43 ± 1.28	90.51 ± 0.19	87.49 ± 0.70	89.81 ± 0.61
GEN [311]	**80.75 ± 0.03**	86.60 ± 0.08	**83.68 ± 0.06**	93.70 ± 0.18	90.25 ± 0.10	90.13 ± 0.06	91.36 ± 0.10
ExCeL [291]	79.39 ± 0.03	85.40 ± 0.04	82.40 ± 0.04	93.76 ± 0.43	92.40 ± 0.05	89.75 ± 0.32	91.97 ± 0.27
Training methods without outliers							
RankOOD (Ours)	**82.57 ± 0.28**	**88.04 ± 0.26**	**85.30 ± 0.18**	**94.69 ± 0.17**	90.85 ± 0.14	**90.89 ± 0.25**	92.14 ± 0.20
CRAFT [290]	80.70 ± 0.18	**86.74 ± 0.84**	**83.65 ± 0.41**	92.85 ± 0.66	89.94 ± 0.49	89.85 ± 0.46	90.88 ± 0.89
ConfBranch [312]	75.01 ± 0.35	83.19 ± 0.14	79.10 ± 0.24	93.40 ± 0.09	89.64 ± 0.52	88.26 ± 0.07	90.43 ± 0.18
G-ODIN [299]	72.94 ± 0.05	81.63 ± 0.21	77.28 ± 0.10	93.12 ± 0.21	**93.67 ± 0.21**	90.18 ± 0.15	**92.33 ± 0.11**
CSI [301]	N/A	N/A	N/A	N/A	N/A	N/A	N/A
ARPL [330]	79.24 ± 0.14	84.81 ± 0.07	82.02 ± 0.10	91.54 ± 0.05	88.11 ± 0.34	88.04 ± 0.20	89.23 ± 0.11
MOS [331]	66.54 ± 0.49	73.14 ± 0.47	69.84 ± 0.46	79.69 ± 1.38	81.38 ± 0.75	80.29 ± 0.68	80.46 ± 0.92
LogitNorm [295]	78.42 ± 0.23	**86.90 ± 0.07**	82.66 ± 0.15	**96.26 ± 0.20**	91.85 ± 0.21	**91.01 ± 0.27**	**93.04 ± 0.21**
CIDER [300]	76.04 ± 2.37	85.13 ± 1.13	80.58 ± 1.75	90.69 ± 2.13	92.38 ± 1.35	88.92 ± 1.58	90.66 ± 1.68
Training methods with outliers							
OE [285]	**82.34 ± 0.16**	**87.35 ± 0.23**	**84.84 ± 0.16**	90.30 ± 0.16	87.76 ± 0.32	89.01 ± 0.24	89.02 ± 0.18
MCD [286]	**81.51 ± 0.14**	85.74 ± 0.07	83.62 ± 0.09	90.83 ± 0.10	86.87 ± 0.12	89.12 ± 0.18	88.94 ± 0.10
UDG [287]	70.73 ± 1.74	77.88 ± 1.56	74.30 ± 1.63	85.95 ± 2.97	81.79 ± 2.57	78.54 ± 2.98	82.09 ± 2.78
MixOE [288]	80.23 ± 0.15	85.01 ± 0.10	82.62 ± 0.03	90.64 ± 0.36	86.80 ± 0.45	87.36 ± 0.49	88.27 ± 0.41

the conditional probability of correctly predicting the rank-9 label increases to 0.92. However, when evaluated on an OOD dataset such as CIFAR-100, this probability decreases to 0.55. This pronounced drop highlights that OOD examples are significantly less likely to preserve the learned ranking structure, thereby providing an effective signal for distinguishing ID from OOD samples.

Table C.6: FPR95 (% ↓) of various methods for different OOD datasets when CIFAR-100 is ID. For each column, the top five methods are marked in **bold**. *RankOOD ranks within the top four methods in four out of six OOD datasets while achieving second best average FPR95 for both near- and far-OOD detection.*

Method	Near OOD			Far OOD				
	CIFAR-10	TIN	Average	MNIST	SVHN	Textures	Places365	Average
Post-hoc inference methods								
OpenMax [289]	60.17 ± 0.97	52.99 ± 0.51	56.58 ± 0.73	53.82 ± 4.74	53.20 ± 1.78	56.12 ± 1.91	**54.85 ± 1.42**	54.50 ± 0.68
MSP [278]	**58.91 ± 0.93**	50.70 ± 0.34	**54.80 ± 0.33**	57.23 ± 4.68	59.07 ± 2.53	61.88 ± 1.28	56.62 ± 0.87	58.70 ± 1.06
TempScale [281]	**58.72 ± 0.81**	50.26 ± 0.16	**54.49 ± 0.48**	56.05 ± 4.61	57.71 ± 2.68	61.56 ± 1.43	56.46 ± 0.94	57.94 ± 1.14
ODIN [324]	60.64 ± 0.56	55.19 ± 0.57	57.91 ± 0.51	45.94 ± 3.29	67.41 ± 3.88	62.37 ± 2.96	59.71 ± 0.92	58.86 ± 0.79
MDS [294]	88.00 ± 0.49	79.05 ± 1.22	83.53 ± 0.60	71.72 ± 2.94	67.21 ± 6.09	70.49 ± 2.48	79.61 ± 0.34	72.26 ± 1.56
MDSEns [294]	95.94 ± 0.16	95.82 ± 0.12	95.88 ± 0.04	**2.83 ± 0.86**	82.57 ± 2.58	84.94 ± 0.83	96.61 ± 0.17	66.74 ± 1.04
RMDS [313]	61.37 ± 0.24	49.56 ± 0.90	55.46 ± 0.41	52.05 ± 6.28	51.65 ± 3.68	53.99 ± 1.06	**53.57 ± 0.43**	52.81 ± 0.63
Gram [325]	92.71 ± 0.64	91.85 ± 0.86	92.28 ± 0.29	53.53 ± 7.45	**20.06 ± 1.96**	89.51 ± 2.54	94.67 ± 0.60	64.44 ± 2.37
EBO [282]	59.21 ± 0.75	52.03 ± 0.50	55.62 ± 0.61	52.62 ± 3.83	53.62 ± 3.14	62.35 ± 2.06	57.75 ± 0.86	56.59 ± 1.38
OpenGAN [326]	78.83 ± 3.94	74.21 ± 1.25	76.52 ± 2.59	63.09 ± 23.3	70.35 ± 2.06	74.77 ± 1.78	73.75 ± 8.32	70.49 ± 7.38
GradNorm [327]	84.30 ± 0.36	86.85 ± 0.62	85.58 ± 0.46	86.97 ± 1.44	69.90 ± 7.94	92.51 ± 0.61	85.32 ± 0.44	83.68 ± 1.92
ReAct [283]	61.30 ± 0.43	51.47 ± 0.47	56.39 ± 0.34	56.04 ± 5.66	50.41 ± 2.02	55.04 ± 0.82	**55.30 ± 0.41**	54.20 ± 1.56
MLS [314]	**59.11 ± 0.64**	51.83 ± 0.70	55.47 ± 0.66	52.95 ± 3.82	53.90 ± 3.04	62.39 ± 2.13	57.68 ± 0.91	56.73 ± 1.33
KLM [314]	84.77 ± 2.95	71.07 ± 0.59	77.92 ± 1.31	73.09 ± 6.67	50.30 ± 7.04	81.80 ± 5.80	81.40 ± 1.58	71.65 ± 2.01
VIM [284]	70.59 ± 0.43	54.66 ± 0.42	62.63 ± 0.27	48.32 ± 1.07	46.22 ± 5.46	**46.86 ± 2.29**	61.57 ± 0.77	**50.74 ± 1.00**
KNN [315]	72.80 ± 0.44	49.65 ± 0.37	61.22 ± 0.14	48.58 ± 4.67	51.75 ± 3.12	**53.56 ± 2.32**	60.70 ± 1.03	53.65 ± 0.28
DICE [298]	60.98 ± 1.10	54.93 ± 0.53	57.95 ± 0.53	51.79 ± 3.67	49.58 ± 3.32	64.23 ± 1.65	59.39 ± 1.25	56.25 ± 0.60
RankFeat [328]	82.78 ± 1.56	78.40 ± 0.95	80.59 ± 1.10	75.01 ± 5.83	58.49 ± 2.30	66.87 ± 3.80	77.42 ± 1.96	69.45 ± 1.01
ASH [297]	68.06 ± 0.44	63.35 ± 0.90	65.71 ± 0.24	66.58 ± 3.88	46.00 ± 2.67	61.27 ± 2.74	62.95 ± 0.99	59.20 ± 2.46
SHE [329]	60.41 ± 0.51	57.74 ± 0.73	59.07 ± 0.25	58.78 ± 2.70	59.15 ± 7.61	73.29 ± 3.22	65.24 ± 0.98	64.12 ± 2.70
GEN [311]	**58.87 ± 0.69**	49.98 ± 0.05	**54.42 ± 0.33**	53.92 ± 5.71	55.45 ± 2.76	61.23 ± 1.40	56.25 ± 1.01	56.71 ± 1.59
ExCeL [291]	61.07 ± 0.81	**49.35 ± 0.31**	55.21 ± 0.56	54.67 ± 5.86	**45.13 ± 0.33**	**51.14 ± 0.14**	58.02 ± 1.28	**52.24 ± 1.90**
Training methods without outliers								
RankOOD (Ours)	**55.71 ± 1.92**	49.47 ± 0.80	**52.59 ± 0.75**	**42.25 ± 2.92**	**39.21 ± 1.59**	**52.88 ± 1.89**	55.40 ± 0.47	**47.44 ± 0.80**
CRAFT [290]	59.19 ± 0.64	**48.26 ± 1.21**	53.73 ± 0.62	48.95 ± 1.90	47.50 ± 5.22	56.97 ± 1.77	**54.02 ± 0.30**	**51.86 ± 0.49**
ConfBranch [312]	74.56 ± 1.22	65.86 ± 0.56	70.21 ± 0.83	55.95 ± 6.15	76.01 ± 12.3	85.43 ± 1.17	69.90 ± 0.28	71.82 ± 3.39
G-ODIN [299]	78.82 ± 1.86	56.34 ± 0.45	67.58 ± 0.98	**27.19 ± 6.24**	**42.68 ± 5.74**	**35.83 ± 1.15**	65.03 ± 1.16	**42.68 ± 3.19**
CSI [301]	72.62 ± 0.49	67.90 ± 0.64	70.26 ± 0.56	80.54 ± 4.87	67.21 ± 3.35	90.51 ± 1.47	69.41 ± 0.58	76.92 ± 1.29
ARPL [330]	64.84 ± 1.25	58.27 ± 2.40	61.56 ± 1.81	59.12 ± 8.04	59.76 ± 1.58	71.66 ± 1.81	62.01 ± 0.89	63.14 ± 2.53
MOS [331]	60.60 ± 1.47	51.49 ± 0.69	56.05 ± 1.01	52.70 ± 3.81	56.33 ± 8.46	61.24 ± 2.06	58.86 ± 0.41	57.28 ± 3.29
LogitNorm [295]	73.88 ± 1.21	51.89 ± 0.10	62.89 ± 0.57	**34.12 ± 8.32**	47.52 ± 8.02	77.38 ± 2.99	55.44 ± 1.45	53.61 ± 3.45
CIDER [300]	82.71 ± 1.25	61.33 ± 0.64	72.02 ± 0.31	75.32 ± 4.21	**17.82 ± 2.80**	54.43 ± 2.56	69.30 ± 1.81	54.22 ± 1.24
Training methods with outliers								
OE [285]	61.26 ± 0.22	**0.21 ± 0.01**	**30.73 ± 0.11**	53.31 ± 9.91	51.84 ± 3.45	55.83 ± 1.82	58.30 ± 0.72	54.82 ± 2.79
MCD [286]	62.65 ± 0.54	**49.10 ± 1.29**	55.88 ± 0.85	62.78 ± 2.91	**43.71 ± 3.73**	56.89 ± 0.64	**54.17 ± 1.13**	54.39 ± 1.34
UDG [287]	66.40 ± 0.51	56.43 ± 0.68	61.42 ± 0.48	**45.14 ± 12.8**	59.67 ± 5.62	71.33 ± 3.59	59.85 ± 0.57	59.00 ± 3.35
MixOE [288]	61.12 ± 1.08	**49.32 ± 0.36**	55.22 ± 0.49	59.49 ± 7.74	73.09 ± 4.00	66.04 ± 0.98	56.93 ± 0.78	63.88 ± 2.48

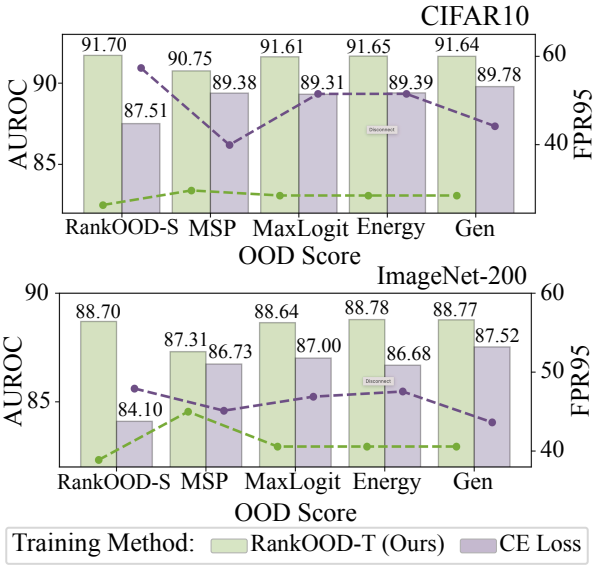


Figure C.1: Avg. AUROC (bars, left y-axis) and FPR95 (lines, right y-axis) across multiple OOD datasets for other scoring functions.

Table C.7: AUROC (% ↑) of various methods for different OOD datasets when CIFAR-100 is ID. For each column, the top five methods are marked in **bold**. *RankOOD ranks within the top four methods in three out of five far-OOD detection scenarios.*

Method	Near OOD			Far OOD				
	CIFAR-10	TIN	Average	MNIST	SVHN	Textures	Places365	Average
Post-hoc inference methods								
OpenMax [289]	74.38 ± 0.37	78.44 ± 0.14	76.41 ± 0.25	76.01 ± 1.39	82.07 ± 1.53	80.56 ± 0.09	79.29 ± 0.40	79.48 ± 0.41
MSP [278]	78.47 ± 0.07	82.07 ± 0.17	80.27 ± 0.11	76.08 ± 1.86	78.42 ± 0.89	77.32 ± 0.71	79.22 ± 0.29	77.76 ± 0.44
TempScale [281]	**79.02 ± 0.06**	82.79 ± 0.09	80.90 ± 0.07	77.27 ± 1.85	79.79 ± 1.05	78.11 ± 0.72	79.80 ± 0.25	78.74 ± 0.51
ODIN [324]	78.18 ± 0.14	81.63 ± 0.08	79.90 ± 0.11	83.79 ± 1.31	74.54 ± 0.76	79.33 ± 1.08	79.45 ± 0.26	79.28 ± 0.21
MDS [294]	55.87 ± 0.22	61.50 ± 0.28	58.69 ± 0.09	67.47 ± 0.81	70.68 ± 6.40	76.26 ± 0.69	63.15 ± 0.49	69.39 ± 1.39
MDSEns [294]	43.85 ± 0.31	48.78 ± 0.19	46.31 ± 0.24	**98.21 ± 0.78**	53.76 ± 1.63	69.75 ± 1.14	42.27 ± 0.73	66.00 ± 0.69
RMDS [313]	77.75 ± 0.19	82.55 ± 0.02	80.15 ± 0.11	79.74 ± 2.49	84.89 ± 1.10	**83.65 ± 0.51**	**83.40 ± 0.46**	**82.92 ± 0.42**
Gram [325]	49.41 ± 0.58	53.91 ± 1.58	51.66 ± 0.77	80.71 ± 4.15	**95.55 ± 0.60**	70.79 ± 1.32	46.38 ± 1.21	73.36 ± 1.08
EBO [282]	**79.05 ± 0.11**	82.76 ± 0.08	**80.91 ± 0.08**	79.18 ± 1.37	82.03 ± 1.74	78.35 ± 0.83	79.52 ± 0.23	79.77 ± 0.61
OpenGAN [326]	63.23 ± 2.44	68.74 ± 2.29	65.98 ± 1.26	68.14 ± 18.8	68.40 ± 2.15	65.84 ± 3.43	69.13 ± 7.08	67.88 ± 7.16
GradNorm [327]	70.32 ± 0.20	69.95 ± 0.79	70.13 ± 0.47	65.35 ± 1.12	76.95 ± 4.73	64.58 ± 0.13	69.69 ± 0.17	69.14 ± 1.05
ReAct [283]	78.65 ± 0.05	82.88 ± 0.08	80.77 ± 0.05	78.37 ± 1.59	83.01 ± 0.97	80.15 ± 0.46	**80.03 ± 0.11**	80.39 ± 0.49
MLS [314]	**79.21 ± 0.10**	82.90 ± 0.05	**81.05 ± 0.07**	78.91 ± 1.47	81.65 ± 1.49	78.39 ± 0.84	79.75 ± 0.24	79.67 ± 0.57
KLM [314]	73.91 ± 0.25	79.22 ± 0.28	76.56 ± 0.25	74.15 ± 2.59	79.34 ± 0.44	75.77 ± 0.45	75.70 ± 0.24	76.24 ± 0.52
VIM [284]	72.21 ± 0.41	77.76 ± 0.16	74.98 ± 0.13	81.89 ± 1.02	83.14 ± 3.71	**85.91 ± 0.78**	75.85 ± 0.37	81.70 ± 0.62
KNN [315]	77.02 ± 0.25	**83.34 ± 0.16**	80.18 ± 0.15	82.36 ± 1.52	84.15 ± 1.09	**83.66 ± 0.83**	79.43 ± 0.47	**82.40 ± 0.17**
DICE [298]	78.04 ± 0.32	80.72 ± 0.30	79.38 ± 0.23	79.86 ± 1.89	84.22 ± 2.00	77.63 ± 0.34	78.33 ± 0.66	80.01 ± 0.18
RankFeat [328]	58.04 ± 2.36	65.72 ± 0.22	61.88 ± 1.28	63.03 ± 3.86	72.14 ± 1.39	69.40 ± 3.08	63.82 ± 1.83	67.10 ± 1.42
ASH [297]	76.48 ± 0.30	79.92 ± 0.20	78.20 ± 0.15	77.23 ± 0.46	**85.60 ± 1.40**	80.72 ± 0.70	78.76 ± 0.16	80.58 ± 0.66
SHE [329]	78.15 ± 0.03	79.74 ± 0.36	78.95 ± 0.18	76.76 ± 1.07	80.97 ± 3.98	73.64 ± 1.28	76.30 ± 0.51	76.92 ± 1.16
GEN [311]	**79.38 ± 0.04**	**83.25 ± 0.13**	**81.31 ± 0.08**	78.29 ± 2.05	81.41 ± 1.50	78.74 ± 0.81	**80.28 ± 0.27**	79.68 ± 0.75
ExCeL [291]	78.14 ± 0.09	**83.26 ± 0.03**	80.70 ± 0.06	78.99 ± 1.73	**85.91 ± 0.73**	**83.28 ± 0.58**	79.98 ± 0.57	**82.04 ± 0.90**
Training methods without outliers								
RankOOD (Ours)	78.84 ± 0.79	82.50 ± 0.39	80.67 ± 0.40	**84.00 ± 2.15**	**87.75 ± 1.25**	82.04 ± 0.70	80.73 ± 0.46	**83.63 ± 1.06**
CRAFT [290]	**78.67 ± 0.21**	83.14 ± 0.73	80.90 ± 0.33	80.34 ± 1.84	85.16 ± 1.15	80.91 ± 0.45	**81.71 ± 0.12**	82.03 ± 0.34
ConfBranch [312]	68.80 ± 0.73	74.41 ± 0.54	71.60 ± 0.62	74.29 ± 4.44	65.51 ± 8.07	65.39 ± 0.16	70.42 ± 0.26	68.90 ± 1.83
G-ODIN [299]	73.04 ± 0.39	81.26 ± 0.29	77.15 ± 0.28	**91.15 ± 2.86**	83.74 ± 3.10	**89.62 ± 0.36**	78.17 ± 0.62	**85.67 ± 1.58**
CSI [301]	69.50 ± 0.43	73.40 ± 0.13	71.45 ± 0.27	51.79 ± 6.77	80.24 ± 1.80	62.22 ± 0.98	70.99 ± 0.54	66.31 ± 1.21
ARPL [330]	73.38 ± 0.78	76.50 ± 1.11	74.94 ± 0.93	73.77 ± 5.89	76.45 ± 1.00	69.93 ± 1.33	74.62 ± 0.57	73.69 ± 1.80
MOS [331]	78.54 ± 0.13	82.26 ± 0.25	80.40 ± 0.18	80.68 ± 1.65	81.59 ± 3.81	79.92 ± 0.57	78.50 ± 0.34	80.17 ± 1.21
LogitNorm [295]	74.57 ± 0.39	82.37 ± 0.24	78.47 ± 0.31	**90.69 ± 1.38**	82.80 ± 4.57	72.37 ± 0.67	**80.25 ± 0.61**	81.53 ± 1.26
CIDER [300]	67.55 ± 0.60	78.65 ± 0.35	73.10 ± 0.39	68.14 ± 3.98	**97.17 ± 0.34**	82.21 ± 1.93	74.43 ± 0.64	80.49 ± 0.68
Training methods with outliers								
OE [285]	76.70 ± 0.19	**99.89 ± 0.02**	**88.30 ± 0.10**	80.68 ± 5.82	84.37 ± 1.34	82.18 ± 0.68	78.39 ± 0.41	81.41 ± 1.49
MCD [286]	75.40 ± 0.46	78.75 ± 0.21	77.07 ± 0.32	68.25 ± 1.99	75.92 ± 0.37	77.07 ± 0.76	77.65 ± 0.09	74.72 ± 0.78
UDG [287]	75.15 ± 0.15	80.90 ± 0.21	78.02 ± 0.10	**83.88 ± 5.98**	79.80 ± 1.61	75.57 ± 0.80	79.11 ± 0.17	79.59 ± 1.77
MixOE [288]	78.17 ± 0.29	**83.73 ± 0.12**	**80.95 ± 0.20**	76.06 ± 5.52	72.28 ± 0.81	77.34 ± 0.91	79.92 ± 0.30	76.40 ± 1.44

Example 01
Given that $\pi^0 = [0, 1, 3, 2]$ and $Ref^0 = [25, 15, -15, -30]$
When predicted logit $x = [15, 7, -1, -3]$
Therefore; $\hat{c} = 0,\ \pi^{\hat{c}} = \pi^0, \bar{\pi} = [0, 1, 2, 3]$,
Assume $w_i = 1$ and $\gamma = 1.1$
$\delta_{\pi_0^{\hat{0}}} = \delta_0 = 1.1^2,\ \ \delta_{\pi_1^{\hat{0}}} = \delta_1 = 1.1^2$
$\delta_{\pi_2^{\hat{0}}} = \delta_3 = 1.1^2,\ \ \delta_{\pi_3^{\hat{0}}} = \delta_2 = 1.1$
$U = [(\frac{15}{(1.1)^2} - 25), (\frac{7}{(1.1)^2} - 15), (\frac{-3}{(1.1)^2} + 15), (\frac{-1}{1.1} + 30)]$
$RankOOD\ Score = -96.57$

Example 02
Given that $\pi^0 = [0, 1, 3, 2]$ and $Ref^0 = [25, 15, -15, -30]$
When predicted logit $x = [27, 7, -10, -1]$
Therefore; $\hat{c} = 0,\ \pi^{\hat{c}} = \pi^0, \bar{\pi} = [0, 1, 3, 2]$,
Assume $w_i = 1$ and $\gamma = 1.1$
Since $\bar{\pi} = \pi^0, \quad \delta_i = \gamma^0 = 1 \forall i \in [0, \ldots, 3]$
$U = [(27 - 25), (7 - 15), (-1 + 15), (-10 + 30)]$
$RankOOD\ Score = 28$

Figure C.2: Toy RankOOD-S computation for a four-class problem.

Table C.8: FPR95 (% ↓) of various methods for different OOD datasets when CIFAR-10 is ID. For each column, the top five methods are marked in **bold**. *RankOOD ranks within the top five methods in two out of three near-OOD detection scenarios.*

Method	Near OOD			Far OOD				
	CIFAR-100	TIN	Average	MNIST	SVHN	Textures	Places365	Average
Post-hoc inference methods								
OpenMax [289]	48.06 ± 3.25	39.18 ± 1.44	43.62 ± 2.27	23.33 ± 4.67	25.40 ± 1.47	31.50 ± 4.05	38.52 ± 2.27	29.69 ± 1.21
MSP [278]	53.08 ± 4.86	43.27 ± 3.00	48.17 ± 3.92	23.64 ± 5.81	25.82 ± 1.64	34.96 ± 4.64	42.47 ± 3.81	31.72 ± 1.84
TempScale [281]	55.81 ± 5.07	46.11 ± 3.63	50.96 ± 4.32	23.53 ± 7.05	26.97 ± 2.65	38.16 ± 5.89	45.27 ± 4.50	33.48 ± 2.39
ODIN [324]	77.00 ± 5.74	75.38 ± 6.42	76.19 ± 6.08	23.83 ± 12.3	68.61 ± 0.52	67.70 ± 11.1	70.36 ± 6.96	57.62 ± 4.24
MDS [294]	52.81 ± 3.62	46.99 ± 4.36	49.90 ± 3.98	27.30 ± 3.55	25.96 ± 2.52	27.94 ± 4.20	47.67 ± 4.54	32.22 ± 3.40
MDSEns [294]	91.87 ± 0.10	92.66 ± 0.42	92.26 ± 0.20	**1.30 ± 0.51**	74.34 ± 1.04	76.07 ± 0.17	94.16 ± 0.33	61.47 ± 0.48
RMDS [313]	43.86 ± 3.49	33.91 ± 1.39	38.89 ± 2.39	21.49 ± 2.32	23.46 ± 1.48	25.25 ± 0.53	31.20 ± 0.28	25.35 ± 0.73
Gram [325]	91.68 ± 2.24	90.06 ± 1.59	90.87 ± 1.91	70.30 ± 8.96	33.91 ± 17.4	94.64 ± 2.71	90.49 ± 1.93	72.34 ± 6.73
EBO [282]	66.60 ± 4.46	56.08 ± 4.83	61.34 ± 4.63	24.99 ± 12.9	35.12 ± 6.11	51.82 ± 6.11	54.85 ± 6.52	41.69 ± 5.32
OpenGAN [326]	94.84 ± 3.83	94.11 ± 4.21	94.48 ± 4.01	79.54 ± 19.7	75.27 ± 26.9	83.95 ± 14.9	95.32 ± 4.45	83.52 ± 11.6
GradNorm [327]	94.54 ± 1.11	94.89 ± 0.60	94.72 ± 0.82	85.41 ± 4.85	91.65 ± 2.42	98.09 ± 0.49	92.46 ± 2.28	91.90 ± 2.23
ReAct [283]	67.40 ± 7.34	59.71 ± 7.31	63.56 ± 7.33	33.77 ± 18.0	50.23 ± 15.9	51.42 ± 11.4	44.20 ± 3.35	44.90 ± 8.37
MLS [314]	66.59 ± 4.44	56.06 ± 4.82	61.32 ± 4.62	25.06 ± 12.9	35.09 ± 6.09	51.73 ± 6.13	54.84 ± 6.51	41.68 ± 5.27
KLM [314]	90.55 ± 5.83	85.18 ± 7.60	87.86 ± 6.37	76.22 ± 12.1	59.47 ± 7.06	81.95 ± 9.95	95.58 ± 2.12	78.31 ± 4.84
VIM [284]	49.19 ± 3.15	40.49 ± 1.55	44.84 ± 2.31	18.36 ± 1.42	19.29 ± 0.41	**21.14 ± 1.83**	41.43 ± 2.17	25.05 ± 0.52
KNN [315]	37.64 ± 0.31	30.37 ± 0.65	34.01 ± 0.38	20.05 ± 1.36	22.60 ± 1.26	24.06 ± 0.55	30.38 ± 0.63	24.27 ± 0.40
DICE [298]	73.71 ± 7.67	66.37 ± 7.68	70.04 ± 7.64	30.83 ± 10.5	36.61 ± 4.74	62.42 ± 4.79	77.19 ± 12.6	51.76 ± 4.42
RankFeat [328]	65.32 ± 3.48	56.44 ± 5.76	60.88 ± 4.60	61.86 ± 12.8	64.49 ± 7.38	59.71 ± 9.79	43.70 ± 7.39	57.44 ± 7.99
ASH [297]	87.31 ± 2.06	86.25 ± 1.58	86.78 ± 1.82	70.00 ± 10.6	83.64 ± 6.48	84.59 ± 1.74	77.89 ± 7.28	79.03 ± 4.22
SHE [329]	81.00 ± 3.42	78.30 ± 3.52	79.65 ± 3.47	42.22 ± 20.6	62.74 ± 4.01	84.60 ± 5.30	76.36 ± 5.32	66.48 ± 5.98
GEN [311]	58.75 ± 3.97	48.59 ± 2.34	53.67 ± 3.14	23.00 ± 7.75	28.14 ± 2.59	40.74 ± 6.61	47.03 ± 3.22	34.73 ± 1.58
ExCeL [291]	71.16 ± 1.34	61.42 ± 0.26	66.55 ± 0.43	**15.46 ± 1.89**	31.78 ± 3.65	53.67 ± 2.19	55.09 ± 1.12	40.03 ± 0.84
Training methods without outliers								
RankOOD (Ours)	**35.42 ± 0.65**	28.01 ± 1.41	**31.72 ± 0.67**	17.18 ± 4.09	15.75 ± 3.20	22.66 ± 5.39	28.27 ± 0.83	20.96 ± 2.55
CRAFT [290]	36.61 ± 2.93	**27.28 ± 0.09**	31.94 ± 1.41	17.13 ± 0.99	14.58 ± 4.62	**20.78 ± 0.04**	**25.12 ± 0.15**	**19.40 ± 0.88**
ConfBranch [312]	**34.44 ± 0.81**	**28.11 ± 0.61**	**31.28 ± 0.66**	**15.79 ± 2.00**	**14.06 ± 0.84**	27.24 ± 1.32	28.85 ± 1.03	21.48 ± 0.94
G-ODIN [299]	48.86 ± 2.91	42.21 ± 2.18	45.54 ± 2.52	**4.53 ± 2.08**	**10.72 ± 0.88**	27.27 ± 6.73	43.30 ± 3.57	21.45 ± 1.91
CSI [301]	37.57 ± 0.89	29.74 ± 0.42	33.66 ± 0.64	24.41 ± 1.57	17.56 ± 0.12	28.95 ± 1.33	34.76 ± 1.52	26.42 ± 0.29
ARPL [330]	43.38 ± 0.37	37.28 ± 1.21	40.33 ± 0.70	21.49 ± 2.03	35.68 ± 3.48	35.19 ± 1.79	37.21 ± 0.80	32.39 ± 0.74
MOS [331]	79.38 ± 5.06	78.05 ± 6.69	78.72 ± 5.86	65.95 ± 17.5	57.79 ± 5.79	76.78 ± 3.86	51.09 ± 1.33	62.90 ± 6.62
LogitNorm [295]	**34.37 ± 1.30**	**24.30 ± 0.54**	**29.34 ± 0.81**	**3.93 ± 1.99**	**8.33 ± 1.78**	**21.94 ± 0.85**	**21.04 ± 0.71**	**13.81 ± 0.20**
CIDER [300]	**35.60 ± 0.78**	28.61 ± 1.10	32.11 ± 0.94	24.76 ± 2.82	**8.04 ± 0.43**	25.05 ± 3.29	**25.03 ± 1.36**	**20.72 ± 0.85**
Training methods with outliers								
OE [285]	36.71 ± 2.06	**2.97 ± 1.17**	**19.84 ± 0.95**	24.67 ± 2.55	**1.25 ± 0.36**	**12.07 ± 2.14**	**14.53 ± 2.80**	**13.13 ± 0.53**
MCD [286]	**34.36 ± 0.37**	**25.98 ± 0.44**	**30.17 ± 0.06**	62.11 ± 11.8	19.43 ± 5.93	22.51 ± 5.16	**24.10 ± 1.58**	32.03 ± 4.21
UDG [287]	40.75 ± 0.69	29.93 ± 1.27	35.34 ± 0.95	16.61 ± 5.14	17.39 ± 7.87	**19.70 ± 1.89**	27.70 ± 1.80	**20.35 ± 2.41**
MixOE [288]	58.29 ± 8.25	44.62 ± 7.57	51.45 ± 7.78	38.28 ± 13.4	20.36 ± 3.99	33.19 ± 4.28	43.54 ± 4.95	33.84 ± 4.77

Table C.9: AUROC (% ↑) of various methods for different OOD datasets when CIFAR-10 is ID. For each column, the top five methods are marked in **bold**. *RankOOD achieves on par performance compared to CRAFT, a class rank based method.*

Method	Near OOD			Far OOD				
	CIFAR-100	TIN	Average	MNIST	SVHN	Textures	Places365	Average
Post-hoc inference methods								
OpenMax [289]	86.91 ± 0.31	88.32 ± 0.28	87.62 ± 0.29	90.50 ± 0.44	89.77 ± 0.45	89.58 ± 0.60	88.63 ± 0.28	89.62 ± 0.19
MSP [278]	87.19 ± 0.33	88.87 ± 0.19	88.03 ± 0.25	92.63 ± 1.57	91.46 ± 0.40	89.89 ± 0.71	88.92 ± 0.47	90.73 ± 0.43
TempScale [281]	87.17 ± 0.40	89.00 ± 0.23	88.09 ± 0.31	93.11 ± 1.77	91.66 ± 0.52	90.01 ± 0.74	89.11 ± 0.52	90.97 ± 0.52
ODIN [324]	82.18 ± 1.87	83.55 ± 1.84	82.87 ± 1.85	95.24 ± 1.96	84.58 ± 0.77	86.94 ± 2.26	85.07 ± 1.24	87.96 ± 0.61
MDS [294]	83.59 ± 2.27	84.81 ± 2.53	84.20 ± 2.40	90.10 ± 2.41	91.18 ± 0.47	92.69 ± 1.06	84.90 ± 2.54	89.72 ± 1.36
MDSEns [294]	61.29 ± 0.23	59.57 ± 0.53	60.43 ± 0.26	**99.17 ± 0.41**	66.56 ± 0.58	77.40 ± 0.28	52.47 ± 0.15	73.90 ± 0.27
RMDS [313]	88.83 ± 0.35	90.76 ± 0.27	89.80 ± 0.28	93.22 ± 0.80	91.84 ± 0.26	92.23 ± 0.23	91.51 ± 0.11	92.20 ± 0.21
Gram [325]	58.33 ± 4.49	58.98 ± 5.19	58.66 ± 4.83	72.64 ± 2.34	91.52 ± 4.45	62.34 ± 8.27	60.44 ± 3.41	71.73 ± 3.20
EBO [282]	86.36 ± 0.58	88.80 ± 0.36	87.58 ± 0.46	94.32 ± 2.53	91.79 ± 0.98	89.47 ± 0.70	89.25 ± 0.78	91.21 ± 0.92
OpenGAN [326]	52.81 ± 7.69	54.62 ± 7.68	53.71 ± 7.68	56.14 ± 24.1	52.81 ± 27.6	56.14 ± 18.3	53.34 ± 5.79	54.61 ± 15.5
GradNorm [327]	54.43 ± 1.59	55.37 ± 0.41	54.90 ± 0.98	63.72 ± 7.37	53.91 ± 6.36	52.07 ± 4.09	60.50 ± 5.33	57.55 ± 3.22
ReAct [283]	85.93 ± 0.83	88.29 ± 0.44	87.11 ± 0.61	92.81 ± 3.03	89.12 ± 3.19	89.38 ± 1.49	90.35 ± 0.78	90.42 ± 1.41
MLS [314]	86.31 ± 0.59	88.72 ± 0.36	87.52 ± 0.47	94.15 ± 2.48	91.69 ± 0.94	89.41 ± 0.71	89.14 ± 0.76	91.10 ± 0.89
KLM [314]	77.89 ± 0.75	80.49 ± 0.85	79.19 ± 0.80	85.00 ± 2.04	84.99 ± 1.18	82.35 ± 0.33	78.37 ± 0.33	82.68 ± 0.21
VIM [284]	87.75 ± 0.28	89.62 ± 0.33	88.68 ± 0.28	94.76 ± 0.38	94.50 ± 0.48	**95.15 ± 0.34**	89.49 ± 0.39	93.48 ± 0.24
KNN [315]	**89.73 ± 0.14**	91.56 ± 0.26	90.64 ± 0.20	94.26 ± 0.38	92.67 ± 0.30	93.16 ± 0.24	**91.77 ± 0.23**	92.96 ± 0.14
DICE [298]	77.01 ± 0.88	79.67 ± 0.87	78.34 ± 0.79	90.37 ± 5.97	90.02 ± 1.77	81.86 ± 2.35	74.67 ± 4.98	84.23 ± 1.89
RankFeat [328]	77.98 ± 2.24	80.94 ± 2.80	79.46 ± 2.52	75.87 ± 5.22	68.15 ± 7.44	73.46 ± 6.49	85.99 ± 3.04	75.87 ± 5.06
ASH [297]	74.11 ± 1.55	76.44 ± 0.61	75.27 ± 1.04	83.16 ± 4.66	73.46 ± 6.41	77.45 ± 2.39	79.89 ± 3.69	78.49 ± 2.58
SHE [329]	80.31 ± 0.69	82.76 ± 0.43	81.54 ± 0.51	90.43 ± 4.76	86.38 ± 1.32	81.57 ± 1.21	82.89 ± 1.22	85.32 ± 1.43
GEN [311]	87.21 ± 0.36	89.20 ± 0.25	88.20 ± 0.30	93.83 ± 2.14	91.97 ± 0.66	90.14 ± 0.76	89.46 ± 0.65	91.35 ± 0.69
ExCeL [291]	85.31 ± 0.26	88.48 ± 0.19	86.89 ± 0.23	**95.87 ± 0.45**	91.40 ± 1.43	89.66 ± 0.64	89.84 ± 0.41	91.69 ± 0.18
Training methods without outliers								
RankOOD (Ours)	89.11 ± 0.24	91.32 ± 0.58	90.21 ± 0.41	93.95 ± 2.23	94.70 ± 1.03	92.63 ± 1.53	91.51 ± 0.36	93.19 ± 0.84
CRAFT [290]	**90.18 ± 0.14**	**92.04 ± 0.06**	**91.11 ± 0.04**	94.59 ± 0.02	94.94 ± 1.10	93.46 ± 0.29	**92.77 ± 0.10**	93.94 ± 0.20
ConfBranch [312]	88.91 ± 0.25	90.77 ± 0.25	89.84 ± 0.24	94.49 ± 0.77	**95.42 ± 0.35**	91.10 ± 0.41	90.39 ± 0.40	92.85 ± 0.29
G-ODIN [299]	88.14 ± 0.60	90.09 ± 0.54	89.12 ± 0.57	**98.95 ± 0.53**	**97.76 ± 0.14**	**95.02 ± 1.10**	90.31 ± 0.65	**95.51 ± 0.31**
CSI [301]	88.16 ± 0.16	90.87 ± 0.23	89.51 ± 0.19	92.55 ± 1.15	95.18 ± 0.45	90.71 ± 0.44	89.56 ± 0.51	92.00 ± 0.30
ARPL [330]	86.76 ± 0.16	88.12 ± 0.14	87.44 ± 0.15	92.62 ± 0.88	87.69 ± 0.97	88.57 ± 0.43	88.39 ± 0.16	89.31 ± 0.32
MOS [331]	70.57 ± 3.04	72.34 ± 3.16	71.45 ± 3.09	74.81 ± 10.1	73.66 ± 9.14	70.35 ± 3.11	86.81 ± 1.85	76.41 ± 5.93
LogitNorm [295]	**90.95 ± 0.22**	**93.70 ± 0.06**	**92.33 ± 0.08**	**99.14 ± 0.45**	**98.25 ± 0.41**	**94.77 ± 0.43**	**94.79 ± 0.16**	**96.74 ± 0.06**
CIDER [300]	89.47 ± 0.19	**91.94 ± 0.19**	**90.71 ± 0.16**	93.30 ± 1.08	**98.06 ± 0.07**	93.71 ± 0.39	**93.77 ± 0.68**	**94.71 ± 0.36**
Training methods with outliers								
OE [285]	**90.54 ± 0.53**	**99.11 ± 0.34**	**94.82 ± 0.21**	90.22 ± 1.31	**99.60 ± 0.14**	**97.58 ± 0.27**	**96.58 ± 0.70**	**96.00 ± 0.13**
MCD [286]	**89.88 ± 0.07**	**92.18 ± 0.18**	**91.03 ± 0.12**	84.22 ± 2.10	93.76 ± 2.30	93.35 ± 1.30	92.66 ± 0.36	91.00 ± 1.10
UDG [287]	88.62 ± 0.32	91.20 ± 0.20	89.91 ± 0.25	**95.81 ± 1.52**	94.55 ± 2.27	**93.92 ± 0.44**	91.97 ± 0.41	**94.06 ± 0.90**
MixOE [288]	87.47 ± 0.97	90.00 ± 0.73	88.73 ± 0.82	91.66 ± 2.21	93.82 ± 1.27	91.84 ± 0.51	90.38 ± 0.55	91.93 ± 0.69

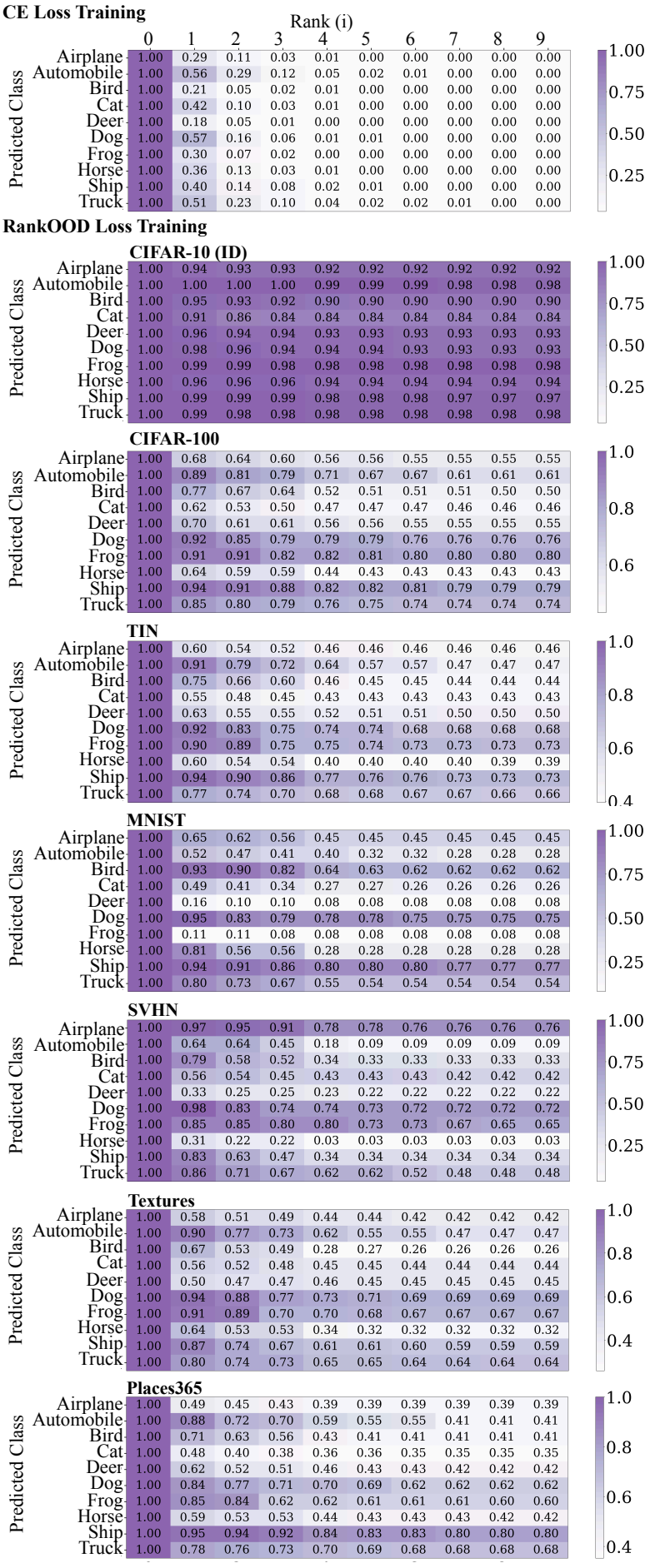


Figure C.3: CIFAR-10 Conditional probability matrix (CP) of rank position i given that all prior ranks have been correctly predicted