TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS

THESIS NO.: THA079MSISE006

MULTI-MODAL HYBRID NEURAL NETWORK FRAMEWORK FOR ENHANCING MOBILE APPLICATION SECURITY AND PRIVACY RISK ANALYSIS

BY

POSHAN KARKI

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR

THE DEGREE OF MASTER OF SCIENCE IN INFORMATICS AND INTELLIGENT SYSTEMS ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING KATHMANDU, NEPAL

NOVEMBER 2025

Multi-Modal Hybrid Neural Network Framework for Enhancing Mobile Application Security and Privacy Risk Analysis

by

Poshan Karki

THA079MSISE006

Thesis Supervisor

Er. Kobid Karkee

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Informatics and Intelligent Systems Engineering

Department of Electronics and Computer Engineering
Institute of Engineering, Thapathali Campus
Tribhuvan University
Kathmandu, Nepal

November 2025

ABSTRACT

Mobile applications frequently request permissions that exceed functional requirements, exposing users to significant privacy risks. Existing security frameworks primarily emphasize malware detection and provide limited mechanisms to evaluate privacy sensitivity or contextual justification of permissions. This research introduces a **Multi-Modal Hybrid Neural Network (MM-HNN)** framework for supervised privacy risk classification of Android applications using app metadata, manifest-declared permissions, genre information, and application descriptions. The proposed framework combines two complementary neural architectures. A **Variational Autoencoder (VAE)** learns compact latent representations of permission and metadata vectors, enhanced with a **classification layer** trained on manually annotated privacy risk labels. In parallel, **DistilBERT** extracts contextual semantic features that capture the alignment between requested permissions and declared app functionality. The system fuses latent structural features and semantic embeddings to build a multimodal classifier capable of identifying privacy-invasive behavior. Trained on a labeled dataset of Nepali-region Android applications, the MM-HNN model is evaluated using standard classification metrics, including accuracy, F1-score, AUC, and confusion matrices. Results demonstrate that the multimodal fusion approach outperforms single-modality baselines by effectively integrating structural anomalies with contextual relevance signals. The framework operates within Android's permission model and Google Play Store guidelines, providing a practical foundation for automated, data-driven mobile privacy risk assessment tools.

Keywords: *Multi-Modal Neural Networks, Variational Autoencoders, Privacy Risk Classification, Permission Analysis, Semantic Alignment, Android Privacy*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Background

Mobile applications have become an essential part of daily life, offering services ranging from messaging and social networking to banking, health monitoring, and entertainment. The widespread adoption of smartphones and app ecosystems has enabled rapid service innovation, but it has also introduced complex privacy and security challenges. Android's permission model is the primary mechanism by which applications request access to sensitive device resources (e.g., location, contacts, camera, microphone) and user data; these permissions are declared in an app's manifest and may be granted at install time or at runtime depending on the permission type and Android version [1, 2, 3]. Prior work has shown that users frequently grant permissions without fully understanding their implications, and developers vary widely in how transparently they justify permission usage [4, 5, 6].

Classical security analyses of mobile apps have focused on detecting malicious behavior through static- or dynamic-analysis techniques and signature- or behavior-based detection systems [7, 8, 9]. While these approaches successfully identify malware and certain classes of runtime misbehavior, they do not directly address privacy risk arising from legitimate applications that request excessive or unjustified permissions. Privacy risk is multi-faceted: it depends not only on which permissions an app requests, but also on the app's stated functionality, developer practices, and how metadata (genre, rating, version history) correlates with permission use [10, 11]. Recent advances in machine learning and natural language processing enable combined analysis of structured metadata and unstructured textual content (e.g., app descriptions, privacy policy excerpts), which can reveal misalignments between requested permissions and declared functionality [12, 13].

Variational Autoencoders (VAEs) and other latent-variable models have demonstrated strong performance for anomaly detection and representation learning in heterogeneous data domains [14, 15], while transformer-based language models (including DistilBERT) provide powerful semantic encodings for short textual documents and descriptions [13, 16]. A multi-modal approach that fuses latent structural representations of permissions and metadata with semantic embeddings from descriptions has the potential to produce more reliable, interpretable, and scalable privacy risk assessments than single-modality systems.

1.2 Motivation

Despite platform-level controls and developer guidance, applications continue to request permissions that are disproportionate to their stated features or user expectations. Large-scale studies and industry reports indicate persistent privacy threats originating from excessive data collection, third-party analytics, and opaque developer practices [17, 18]. Manual review of millions of apps is infeasible for marketplaces and regulators, and most existing automated tools are optimized for malware detection rather than nuanced privacy evaluation [7, 10].

There is a practical and academic need for automated tools that reason about the contextual justification for requested permissions (i.e., do the permissions match stated functionality?), detect structural anomalies across permission and metadata vectors, and present interpretable outputs suitable for stakeholders (users, developers, enterprises, and regulators). Combining representation learning for permission structures (to identify anomalous or rare permission profiles) with semantic analysis (to verify description-permission alignment) addresses this gap and improves the fidelity of privacy risk classification in real-world app ecosystems [12, 13].

Additionally, regional app ecosystems (including Nepali-centered apps) may exhibit distinct patterns of permission usage and metadata characteristics that are underrepresented in global datasets. Building and validating a labeled dataset that reflects local app behavior strengthens the relevance and fairness of any automated privacy assessment tool.

1.3 Problem Statement

Applications frequently request permissions that exceed operational requirements or lack clear justification in their descriptions and metadata, creating privacy risks that are not captured by malware-focused detectors or simple permission-count heuristics. Current automated analyses generally fall short in one or more of the following aspects:

- Reliance on single-modality analysis (either structural or textual) that misses cross-modal inconsistencies [10, 12].
- Lack of labelled datasets that codify privacy risk levels for supervised evaluation, especially for regionally focused applications [19, 20].
- Limited interpretability for stakeholders who must decide whether permission requests are justified or risky under regulatory frameworks such as GDPR and CCPA [21, 22].

This thesis formulates the problem as supervised privacy risk classification: given an Android application's manifest-declared permissions, metadata, and textual description, predict a privacy risk label that reflects the likelihood of unjustified or excessive data access. The challenge lies in designing a multimodal model that captures latent structural anomalies in permission vectors, understands semantic alignment between descriptions and permissions, and fuses these signals into a robust, interpretable classifier suitable for evaluation and deployment.

1.4 Objectives

The objectives of this thesis are:

1. To construct and validate a manually labeled dataset of Android applications (with regional representation).
2. To develop a supervised multimodal hybrid neural network (MM-HNN) to predict privacy risk levels.

1.5 Scope

The scope of this research is defined as follows:

- **Data sources and modalities:** The study focuses on static, publicly available app metadata — manifest-declared permissions, app descriptions, genre, developer information, content ratings, and other Play Store metadata accessible without dynamic instrumentation [1, 23].

- **Modeling approach:** The thesis develops and evaluates a supervised multimodal model combining a VAE for structural feature learning and a DistilBERT encoder for semantic feature extraction. Fusion strategies (e.g., concatenation + classifier, late fusion) and classification head designs are explored and compared.

- **Evaluation:** Performance is evaluated using labeled data and standard classification metrics, including ablation studies to measure the contribution of each modality.

- **Excluded topics:** The thesis does not perform dynamic runtime analysis (network traffic inspection, dynamic taint tracking), on-device instrumentation, or active privacy policy scraping beyond publicly available descriptions and metadata. Detection or mitigation of adversarial evasion techniques is considered out of scope for the core contribution.

1.6 Potential Applications

The MM-HNN framework can serve multiple stakeholders in the mobile ecosystem:

- **End users:** Provide clearer, data-driven privacy risk indicators at install-time or in app marketplaces to inform user consent [24].

- **App stores and marketplace operators:** Automate preliminary privacy vetting and flagging of apps whose permission requests are inconsistent with descriptions or industry norms [25, 23].

- **Developers:** Offer pre-submission privacy checks that suggest permission minimization or improved documentation to reduce rejection risk and enhance user trust [26, 27].

- **Enterprises and IT administrators:** Screen third-party apps for deployment on corporate devices to limit privacy and compliance exposure.

- **Regulators and auditors:** Support large-scale compliance assessments with privacy regulations such as GDPR and CCPA by surfacing apps with potentially non-compliant permission practices [21, 22].

- **Security researchers:** Enable systematic study of privacy trends and longitudinal analysis of permission misuse across categories and regions [12, 17].

1.7 Originality of the Thesis

This thesis introduces a **supervised, multi-modal, and data-driven framework** for assessing privacy risks in Android applications. The originality of this work lies in several key aspects:

1. **Manually Labeled Privacy Risk Dataset:** A primary contribution is the creation of a validated dataset of Android applications annotated with privacy risk labels. This dataset enables supervised training and provides a benchmark for future research in mobile privacy assessment.
2. **Supervised Multi-Modal Learning:** The proposed MM-HNN framework integrates **Variational Autoencoders (VAE)** to extract structured features from app metadata and permissions, along with **DistilBERT** embeddings for semantic analysis of textual descriptions and privacy policies. A classification layer combines these modalities to predict discrete privacy risk levels.
3. **Risk Classification and Confidence Scores:** Unlike purely anomaly-based methods, this framework produces both privacy risk classes and associated confidence scores, offering actionable insights for users, developers, and regulators.
4. **Scalability and Practical Applicability:** By operating on static app information, the framework can efficiently evaluate large-scale datasets while maintaining the reliability of predictions through supervised learning on manually labeled data.

In summary, this research contributes a **novel, practical, and supervised approach** to Android privacy risk assessment by combining manual labeling, multi-modal feature extraction, and classification-based risk prediction. This approach addresses gaps in existing methodologies and establishes a foundation for automated, large-scale privacy evaluation in mobile ecosystems.

1.8 Organization of the Report

This thesis is structured into the following chapters:

- **Chapter 1: Introduction** – This chapter provides an overview of the research background, motivation, problem statement, objectives, scope, potential applications, and the originality of the study.
- **Chapter 2: Literature Review** – It examines existing approaches in mobile application security, highlighting their limitations and identifying the research gaps that the proposed framework addresses.
- **Chapter 3: Methodology** – This chapter details the design, implementation, and technical aspects of the MM-HNN framework, including data collection, model training, and evaluation techniques.
- **Chapter 4: Results** – This chapter presents the evaluation of the framework, including performance metrics, training outcomes, and analyses of both optimal and worst-case scenarios.
- **Chapter 5: Discussion and Analysis** – This chapter provides an in-depth interpretation of the results, discusses their implications, and compares the findings with existing approaches.
- **Chapter 6: Conclusion and Future Work** – This chapter summarizes the key research findings, discusses the contributions of the study, and suggests potential areas for future exploration and enhancements.

2 LITERATURE REVIEW

The exponential growth of mobile applications has reshaped the digital ecosystem, enabling high user engagement but simultaneously introducing severe security and privacy challenges. Traditional defenses—such as signature-based antivirus tools, heuristic scanners, and basic static analysis—are insufficient against modern polymorphic malware, obfuscation techniques, and subtle permission abuse. Consequently, research has shifted toward machine learning (ML), deep learning (DL), and, more recently, Large Language Models (LLMs) for analyzing app behavior, metadata, permissions, and user-facing descriptions. This chapter reviews these advancements across four domains: LLMs, multimodal deep learning, graph-based methods, and conventional ML approaches. It concludes by identifying research gaps that motivate the proposed MM-HNN framework.

2.1 Large Language Models (LLMs) in Mobile Security

Large Language Models (LLMs) have gained significant attention due to their ability to understand and generate structured code and natural language. Chen et al. [28] demonstrated that Codex—an LLM fine-tuned on large code corpora—can infer program structure and logic with high accuracy. Similarly, Khare et al. [29] evaluated 16 LLMs on multiple vulnerability datasets, revealing substantial differences in detection performance across vulnerability categories. Liu et al. [30] introduced VulDetectBench, a standardized benchmark for evaluating LLMs in vulnerability detection, and found that coarse-grained reasoning is handled reasonably well by LLMs, whereas fine-grained semantic vulnerabilities remain a challenge.

While these studies show the potential of LLMs in security analysis, their application to mobile privacy risk—particularly permission–functionality alignment—remains under-explored. Moreover, Pearce et al. [31] revealed that LLM-generated code often contains vulnerabilities, highlighting risks associated with relying solely on automated semantic models.

2.1.1 LLM Deployment Constraints in Mobile Contexts

Despite their capabilities, LLMs remain difficult to deploy directly on mobile devices. Prior work highlights their dependency on high memory capacity, substantial computation, and stable network access [32]. Even lighter variants introduce non-trivial energy consumption and inference latency, making real-time or user-facing deployment impractical. Additionally, cloud-based inference may violate privacy requirements when analyzing sensitive mobile data.

This thesis does not attempt on-device LLM deployment or latency optimization; instead, LLM deployment challenges are discussed to contextualize the design choice of using DistilBERT within a server-side semantic analysis pipeline.

2.1.2 Contextual Understanding and Explainability

Kouliaridis et al. [32] assessed LLMs such as GPT-4 for Android vulnerability detection using extended contextual information, achieving 89.3% accuracy but noting latency and privacy limitations. Zhong et al. [33] introduced a hybrid detection model combining generative architectures with gradient-based explainability (XAI), achieving 93.1% accuracy. However, the computational cost of gradient attribution limits its applicability in mobile-facing tools.

2.1.3 Hybrid Static–Dynamic LLM Methods

Mathews et al. [34] proposed *LLbezpeky*, blending static analysis with LLM-driven dynamic test case generation. While they achieved a 91.67% detection rate on the Ghera benchmark, obfuscation caused a notable false-positive increase, suggesting weaknesses in semantic generalization. Yang et al. [35] explored vulnerabilities in multimodal agents processing text, image, and audio inputs, reaching 87% accuracy. Nevertheless, these multimodal threat vectors differ substantially from permission–functionality alignment in privacy assessments.

Overall, LLMs provide strong semantic reasoning but are constrained by their computational demands, cloud reliance, and lack of mobile privacy-focused datasets. These limitations reinforce the need for hybrid, multi-modal approaches that balance semantic depth with model efficiency.

2.2 Multimodal and Deep Learning Approaches

Deep learning has enabled sophisticated fusion of heterogeneous features extracted from Android applications. Park et al. [12] introduced *MultiVulnDet*, combining

CNN-based binary visualization with Transformer-based code sequence analysis, achieving 89.5% accuracy. However, the reliance on large annotated datasets and high computational requirements limits real-world adoption.

Chen et al. [5] developed *DeepVuln*, a Bi-LSTM model with attention mechanisms, scoring 87.2% accuracy. Although effective for sequential patterns, Bi-LSTMs struggle with long-range dependencies and unseen vulnerabilities. Wang et al. [36] proposed *VulnHunter*, focusing on third-party library analysis. Despite achieving 86.4% accuracy, its design is optimized for backend infrastructure rather than privacy assessment for end users.

Most multimodal systems target malware detection rather than evaluating excessive permission requests or detecting semantic inconsistencies between app descriptions and permissions—a critical limitation addressed by the MM-HNN framework.

2.3 Graph-Based and Conventional Machine Learning Techniques

Before the rise of LLMs and deep multimodal models, graph-based and traditional ML techniques dominated Android security research.

2.3.1 Graph Neural Networks (GNNs)

Zhang and Roberts [37] introduced *SecureFlow*, which models Android applications through call graphs and data-flow graphs. Using GNNs, they achieved 88.7% accuracy in vulnerability detection. However, building and processing large-scale graphs require substantial computational resources, limiting suitability for lightweight or user-facing applications.

2.3.2 Ensemble and Traditional ML

Traditional ML methods provide lower computational overhead but lack semantic understanding. Kumar et al. [38] built a hybrid ensemble model combining neural networks and decision trees, achieving 89% accuracy. Their earlier work [39] optimized Random Forest classifiers through manual feature engineering, but these models treat permissions as flat features without contextual reasoning.

For example, traditional ML can detect that a calculator app requesting GPS permission is unusual, but it cannot determine whether such permission aligns with the app's stated purpose. This semantic gap motivates the need for multi-modal models that incorporate textual context—such as app descriptions and privacy policies.

2.4 Critical Analysis and Research Gaps

A review of existing literature reveals four major gaps:

1. **Lack of Semantic Permission–Functionality Alignment:** Most prior systems analyze permissions or behavior but fail to evaluate whether requested permissions are justified by the app's declared functionality.
2. **Efficiency–Accuracy Imbalance:** High-accuracy LLM and deep multi-modal systems are computationally expensive, whereas lightweight models lack contextual reasoning.
3. **Limited Interpretability:** Current deep learning models often function as black boxes. Users need understandable, actionable explanations such as "Location permission inconsistent with a wallpapers app," rather than opaque probability outputs.
4. **Restricted Runtime Monitoring:** Android OS limitations hinder dynamic monitoring for non-rooted users, requiring analysis of static artifacts such as metadata, descriptions, and permission lists.

2.5 Critical Analysis and Research Gaps

The literature reveals a distinct divide between highly accurate yet computationally expensive models and lightweight yet contextually limited models. The following research gaps persist:

2.6 Positioning the MM-HNN Framework

The proposed MM-HNN framework addresses these gaps through a hybrid multi-modal architecture combining structured permission analysis with semantic understanding.

- **Variational Autoencoder (VAE) for Structured Features:** The VAE models permission distributions and metadata, identifying anomalies in permission–category relationships with low computational overhead.

- **DistilBERT for Semantic Understanding:** DistilBERT captures contextual information from app descriptions and privacy policies, enabling the framework to analyze whether permission requests align with stated functionality.

- **Supervised Risk Classification:** The fusion of structured and semantic features enables MM-HNN to classify applications into risk categories aligned with privacy assessment requirements and regulatory compliance.

Through this multi-modal fusion, MM-HNN provides a balanced approach that achieves semantic depth without imposing the computational or deployment constraints of large LLM-based systems.

3 METHODOLOGY

3.1 Theoretical Formulations

This section presents the theoretical foundation of the proposed Multi-Modal Hybrid Neural Network (MM-HNN) used for privacy risk classification of Android applications. The framework integrates structural, semantic, and contextual information obtained from permissions, metadata vectors, genre , and application descriptions. The model consists of three major components: (i) a Variational Autoencoder (VAE) for structural risk representation, (ii) a DistilBERT-based contextual encoder, and (iii) a multimodal fusion classifier.

3.1.1 Basic Concept of the Chosen Model

The Variational Autoencoder (VAE) is employed to learn a compact latent representation of high-dimensional permission and metadata vectors. VAEs provide a probabilistic generative framework that captures the underlying structure of the input distribution while encouraging smooth latent manifolds. This structural representation is crucial for detecting abnormal or excessive permission combinations.

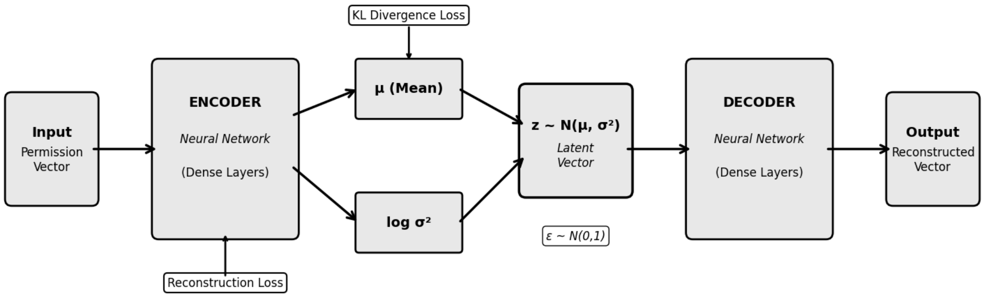

Figure 3.1: Variational Autoencoder Architecture

Parallel to the VAE, DistilBERT is utilized to extract contextual and semantic features from the application descriptions. DistilBERT preserves the linguistic reasoning capability of BERT while being computationally efficient, making it suitable for large-scale app analysis. It produces sentence-level embeddings that represent the functional intent of the application.

The MM-HNN framework fuses both modalities—structural latent features and

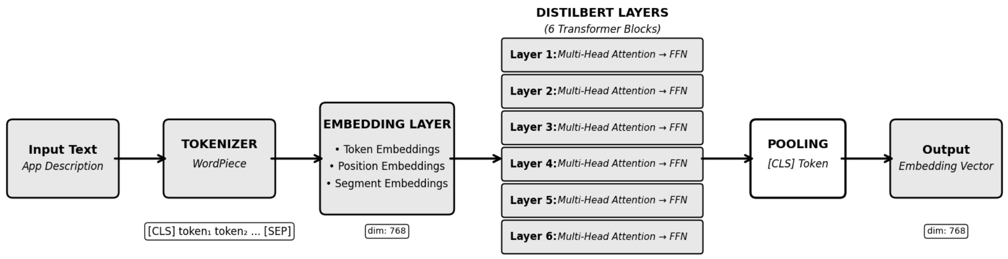
DISTILBERT LAYERS
(6 Transformer Blocks)
Layer 1: Multi-Head Attention → FFN
Layer 2: Multi-Head Attention → FFN
Layer 3: Multi-Head Attention → FFN
Layer 4: Multi-Head Attention → FFN
Layer 5: Multi-Head Attention → FFN
Layer 6: Multi-Head Attention → FFN
Input Text
App Description
TOKENIZER
WordPiece
[CLS] token₁ token₂ ... [SEP]
EMBEDDING LAYER
• Token Embeddings
• Position Embeddings
• Segment Embeddings
dim: 768
POOLING
[CLS] Token
Output
Embedding Vector
dim: 768

Figure 3.2: DistilBERT Architecture

semantic embeddings—along with a manually constructed genre-weight feature to produce a unified risk classification. This multimodal approach enables the system to jointly evaluate the necessity, justification, and sensitivity of permissions requested by the application.

3.1.2 Major Benefits of the Chosen Techniques

- **Variational Autoencoder:**
  - Learns smooth and continuous latent representations suitable for anomaly detection.
  - Performs dimensionality reduction while preserving structural relationships among permissions.
  - Captures latent privacy risk patterns beyond raw permission counts.
- **DistilBERT Encoder:**
  - Extracts deep contextual information from app descriptions.
  - Captures semantic consistency between app purpose and requested permissions.
  - Lightweight compared to BERT, reducing computational complexity.
- **Multimodal Fusion:**
  - Integrates structural anomalies with contextual meaning.
  - Provides more robust classification than single-modality models.

  - Enables detection of over-privileged apps even when permissions seem normal but contextually unjustified.

3.1.3 Assumptions Considered

- The manifest-declared permissions reflect the functional behavior of the application.
- App descriptions provided in the Google Play Store represent the developer's declared intent.
- The dataset of Nepali-region Android apps is representative of the general permission–usage patterns.
- Permissions are treated as binary features (requested or not requested).
- Genre information is assumed to influence the permission justification (e.g., health apps may reasonably request sensors).
- Labeled risk categories (Low, Moderate, High, Critical) accurately reflect expert-annotated privacy sensitivity.

3.2 Mathematical Modeling

1. **Preprocessing of Raw Data:** Raw features include permissions, metadata, descriptions, and genre information. These are transformed into ML-ready vectors.

   (a) *Binary Encoding of Permissions:* Each manifest-declared permission is encoded as a binary feature:

   $$p_j = \begin{cases} 1, & \text{if permission } j \text{ is requested} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

   where $j = 1, 2, \ldots, P$ and P is the total number of permission types.

(b) *Metadata Normalization:* Continuous metadata features (e.g., downloads, ratings) are normalized:

$$\hat{m}_k = \frac{m_k - \min(m_k)}{\max(m_k) - \min(m_k)} \tag{3.2}$$

where $k = 1, 2, \ldots, K$ and K is the total number of metadata features.

(c) *Genre Weighting:* Each app genre is assigned a sensitivity weight:

$$g_i = \text{predefined sensitivity weight for app } i \tag{3.3}$$

where $i = 1, 2, \ldots, N$ and N is the total number of apps.

(d) *Structural Feature Vector:* Combine permissions, normalized metadata, and genre weight:

$$\mathbf{x}_{struct} = [\mathbf{p}, \hat{\mathbf{m}}, g] \tag{3.4}$$

The structural vector lies in:

$$\mathbf{x}_{struct} \in \mathbb{R}^{P+K+1} \tag{3.5}$$

2. **VAE Module: Structural Feature Modeling** The VAE maps $\mathbf{x}_{struct}$ to a latent representation $\mathbf{z}$.

(a) *Encoder:* Produces mean and log-variance vectors:

$$\mu = f_\mu(\mathbf{x}_{struct}), \qquad \log \sigma^2 = f_{\log \varphi}(\mathbf{x}_{struct}) \tag{3.6}$$

(b) *Reparameterization Trick:* Samples latent vector $\mathbf{z}$:

$$\mathbf{z} = \mu + \epsilon \cdot \sigma, \qquad \epsilon \sim \mathcal{N}(0, I) \tag{3.7}$$

(c) *Decoder:* Reconstructs input vector:

$$\hat{\mathbf{x}}_{struct} = f_{dec}(\mathbf{z}) \tag{3.8}$$

(d) *VAE Loss Function:*

$$\mathcal{L}_{VAE} = \sum_i x_i \log \hat{x}_i + (1 - x_i) \log(1 - \hat{x}_i) + \beta \left(-\frac{1}{2} \sum_j (1 + \log \sigma_j^2 - \mu_j^2 - \sigma_j^2) \right) \tag{3.9}$$

3. **DistilBERT Semantic Module** Converts tokenized app descriptions into semantic embeddings:

$$\mathbf{e}_{bert} = \text{DistilBERT}(\text{tokenized description}) \tag{3.10}$$

4. **Fusion Layer** Combines structural latent, semantic embedding, and genre weight:

$$\mathbf{h}_{fusion} = [\mathbf{z}, \mathbf{e}_{bert}, g] \tag{3.11}$$

The fusion network outputs logits:

$$\mathbf{o} = f_{fusion}(\mathbf{h}_{fusion}) \tag{3.12}$$

5. **Post-Processing**

(a) Convert logits to probabilities:

$$\hat{y}_c = \frac{\exp(o_c)}{\sum_{i=1}^{4} \exp(o_i)}, \qquad c = 1, 2, 3, 4 \tag{3.13}$$

(b) Predicted risk category:

$$\text{Predicted Risk} = \arg\max_c \hat{y}_c \tag{3.14}$$

3.3 System Block Diagram

Figure 3.3 illustrates the overall workflow of the proposed Multi-Modal Hybrid Neural Network (MM-HNN) for Android application privacy risk classification. The system integrates preprocessing, structural and semantic feature extraction, multimodal fusion, and final risk prediction in a cohesive pipeline.

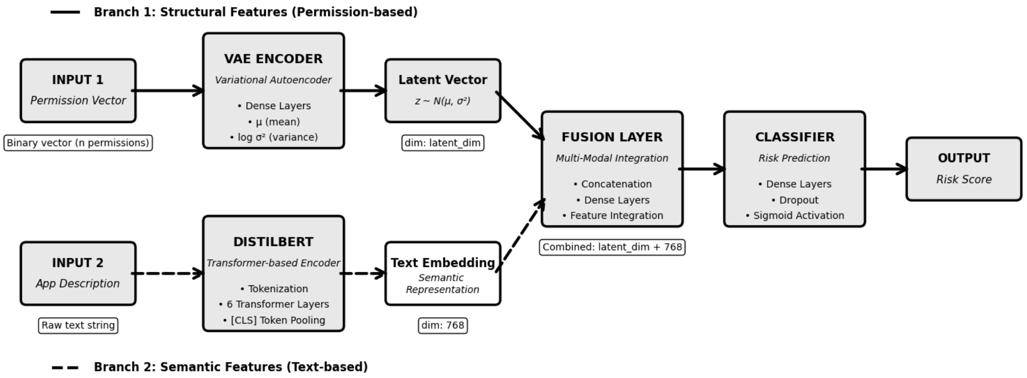

Figure 3.3: System Block DIagram of MMHNN Architecture

3.3.1 Input Stage

The input to the system consists of a labeled dataset of Android applications, including:

- Manifest-declared permissions
- Application metadata (install count, size, content rating, etc.)
- Application description text
- Genre information

During preprocessing, binary encoding is applied to permission features, continuous metadata is normalized, genre information is transformed into a scalar weight, and application descriptions are tokenized for text embedding.

3.3.2 Intermediate Stages

1. **Structural Module (VAE)** The Variational Autoencoder (VAE) encodes the high-dimensional structural input vector into a latent representation $\mathbf{z}$. During training, the decoder reconstructs the input to enforce a smooth latent manifold. The latent vector captures structural privacy risk patterns, including abnormal or excessive permission combinations.

2. **Semantic Module (DistilBERT)** DistilBERT processes the tokenized application descriptions to produce contextual embeddings $\mathbf{e}_{bert}$. These

embeddings capture semantic consistency between declared app functionality and requested permissions, providing complementary information to the structural features.

3. **Fusion Layer**

   The latent vector $\mathbf{z}$, semantic embedding $\mathbf{e}_{bert}$, and genre weight g are concatenated to form a combined feature vector. This vector passes through multiple dense layers with ReLU activation and dropout, allowing the model to learn complex interactions between structural and contextual information.

4. **Output Stage**

   The fusion classifier outputs logits corresponding to four risk categories: *Low*, *Moderate*, *High*, and *Critical*. During training, both the VAE reconstruction/KL loss and the supervised cross-entropy classification loss are optimized. The system is evaluated using standard metrics including accuracy, F1-score, AUC, and confusion matrices.

3.3.3 Training, Validation, and Testing Phases

- **Training Phase:** End-to-end backpropagation with joint VAE and classifier losses to learn structural and semantic representations.
- **Validation Phase:** Forward pass through the network to evaluate intermediate performance and tune hyperparameters.
- **Testing Phase:** Forward pass for final privacy risk predictions, with evaluation using the chosen metrics.

3.4 Instrumentation Usage

3.4.1 Hardware Tools

The project utilized both local and remote computational resources:

- **Development Environment:** Local systems with adequate memory, multi-core processors, and GPU acceleration for prototyping and debugging.

- **High-Performance Computing:** Remote GPU servers for computationally intensive training, reducing training time and enabling large-scale experimentation.
- **Testing Devices:** Multiple Android devices and emulators for testing application functionality, compatibility, and performance across various operating environments.

3.4.2 Software Tools

A range of software frameworks and utilities supported model design, data handling, and visualization:

- **Model Development:** Python-based deep learning frameworks (TensorFlow, PyTorch) for implementing VAE and DistilBERT components.
- **Data Processing and Visualization:** Python libraries for automated webscraping and preprocessing pipelines; Matplotlib and Plotly for interpreting model performance metrics.
- **Reproducibility and Deployment:** Docker for consistent execution across environments; cloud service interfaces for scalability and remote experimentation.

3.5 Dataset Overview and Relevance

Android applications represent a critical security challenge in the mobile ecosystem, with over 3 million applications available on the Google Play Store [40]. While existing research has focused primarily on binary malware classification [41], there exists a significant gap in nuanced risk assessment that considers the spectrum of privacy and security concerns present in legitimate applications. Our research addresses this gap by constructing a multi-dimensional dataset that captures both *structural* (permission-based) and *contextual* (description-based) risk indicators.

3.5.1 Dataset Relevance

The constructed dataset is specifically relevant for the following reasons:

1. **Granular Risk Assessment**: Unlike binary malware datasets (e.g., Drebin [41], AndroZoo [42]), our dataset provides three-level risk categorization (Low, Medium, High), enabling fine-grained security analysis.

2. **Multimodal Features**: The dataset integrates permissions, genre information, textual descriptions, and metadata, supporting multimodal machine learning approaches that leverage complementary signals.

3. **Real-World Applicability**: Data collected from the Google Play Store represents actual applications accessible to end-users, ensuring ecological validity for practical deployment.

4. **Privacy-Focused**: Emphasizes privacy and permission anomalies rather than solely malicious code, addressing the growing concern of data harvesting by legitimate applications [43].

3.5.2 Dataset Statistics

The final validated dataset comprises **3,529 Android applications** collected from the Google Play Store, with the following characteristics:

Table 3.1: Dataset Composition and Statistics

Characteristic	**Value**
Total Applications Collected	4,141
High-Confidence Samples	3,529
Unique Permissions	166
Unique Genres	13
Training Set	2,469 (70.0%)
Validation Set	530 (15.0%)
Test Set	530 (15.0%)

Table 3.2: Risk Label Distribution (Initial Collection)

Label	**Value**
Low Risk	1,735 (41.9%)
Medium Risk	1,578 (38.1%)
High Risk	828 (20.0%)
Total	**4,141 (100%)**

Table 3.3: Risk Label Distribution (High-Confidence Dataset)

Label	**Value**
Low Risk	1,707 (48.4%)
Medium Risk	938 (26.6%)
High Risk	884 (25.0%)
Total	**3,529 (100%)**

3.5.3 Dataset Collection Methodology

Data collection was performed through automated scraping of publicly available metadata from the Google Play Store. The procedure adheres to ethical guidelines and terms of service, collecting only public information without requiring user authentication or accessing personal data.

1. **Collection Pipeline**

   The data collection pipeline consists of four stages:

   (a) **Genre-Stratified Sampling**: Applications were sampled across 13 major genre categories to ensure diverse representation (Table 3.4).

   (b) **Metadata Extraction**: For each application, the following metadata was collected:

   - Package name and version information
   - Application title and description
   - Genre classification

    - Permission declarations
    - Download counts and user ratings
    - Developer information
    - Content rating
    - Pricing information

   (c) **Quality Filtering**: Applications with incomplete metadata (e.g., missing descriptions or permission information) were excluded.

   (d) **Temporal Consistency**: All data was collected within a 30-day window (November 2025) to ensure temporal consistency and minimize version-related discrepancies.

2. **Automated Collection Framework** The collection framework was implemented in Python using the following components:

   - `google-play-scraper` library for metadata retrieval
   - Rate-limiting mechanisms (1 request per second) to respect server constraints
   - Retry logic with exponential backoff for handling transient failures
   - JSON-based storage for structured data persistence

3.5.4 Genre Distribution

Applications were sampled from 13 genre categories based on Google Play Store classifications. Table 3.4 presents the distribution across genre categories, demonstrating balanced representation of application types.

3.5.5 Data Preprocessing and Feature Engineering

3.5.5.1 Permission Processing

Android permissions represent a critical security signal. The preprocessing pipeline addresses the heterogeneity and verbosity of permission declarations through a three-stage process.

1. **Permission Vocabulary Construction**

Table 3.4: Genre Distribution in Dataset

Genre Category	**Count**	**Percentage**
Gaming	1,755	42.4%
Health & Fitness	720	17.4%
Social & Communication	341	8.2%
Lifestyle & Personal	277	6.7%
Entertainment & Media	224	5.4%
Productivity & Work	188	4.5%
Education & Knowledge	172	4.2%
Utilities & System	132	3.2%
Finance	109	2.6%
Shopping & Commerce	105	2.5%
News & Magazines	50	1.2%
Navigation & Mobility	42	1.0%
Kids & Family	26	0.6%
Total	**4,141**	**100%**

After normalization and deduplication, the final permission vocabulary consists of **166 unique permissions**. This vocabulary captures all permission declarations across the 4,141 collected applications.

2. **Permission Categorization**

   Raw permissions are first categorized into eight high-level categories based on Android security model:

   - **Location**: GPS-based and network-based location access
   - **Storage**: Read/write access to external storage
   - **Camera**: Image and video capture capabilities
   - **Microphone**: Audio recording permissions
   - **Contacts**: Access to contact database
   - **Phone**: Call initiation and phone state access
   - **SMS**: Send/receive SMS messages
   - **Other**: Network, Bluetooth, and miscellaneous permissions

3. **Permission Normalization**

Permission strings exhibit significant variability in their textual representation. We implemented a normalization procedure that maps natural language descriptions to standardized Android permission constants. Examples include:

- "*precise location (GPS and network-based)*" → `ACCESS_FINE_LOCATION`
- "*take pictures and videos*" → `CAMERA`
- "*read the contents of your USB storage*" → `READ_EXTERNAL_STORAGE`

4. **Permission Encoding**

   The hierarchical permission structure is flattened into a binary vector representation. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_{166}\}$ be the set of all unique permissions in the dataset. For application i, the permission vector is defined as:

$$\mathbf{p}_i = [p_{i,1}, p_{i,2}, \ldots, p_{i,166}] \in \{0,1\}^{166} \tag{3.15}$$

   where $p_{i,j} = 1$ if application i requests permission p_j, and $p_{i,j} = 0$ otherwise.

3.5.5.2 Text Feature Processing

Textual features (application title and description) undergo the following preprocessing:

1. **Concatenation**: Title and description are concatenated to form a unified text representation.
2. **Text Dimension**: The text feature matrix has dimensions $3529 \times d_{\text{text}}$, where d_{text} represents the embedding dimension used by the transformer model.
3. **Tokenization**: Text is tokenized using DistilBERT's WordPiece tokenizer with maximum sequence length of 256 tokens, ensuring coverage of full descriptions.

Figure 3.4 shows the distribution of description lengths across the dataset.

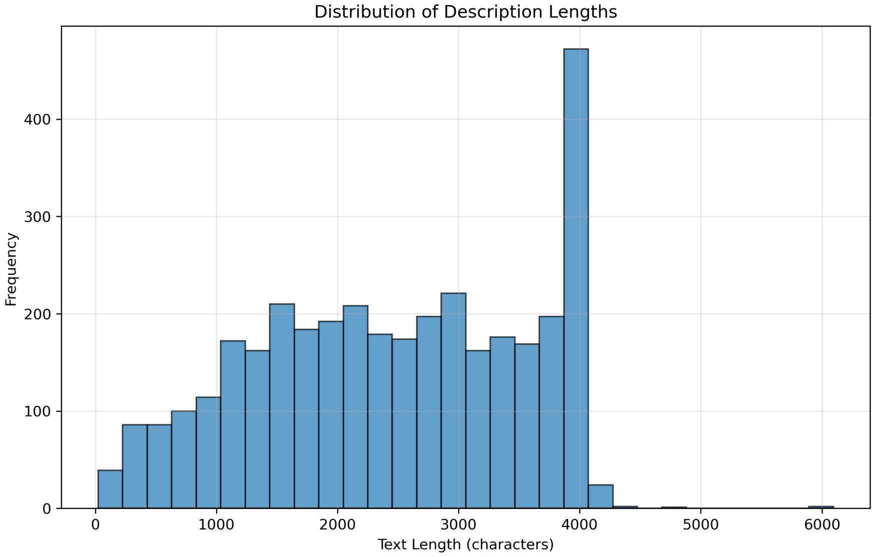

Figure 3.4: Distribution of text description lengths

3.5.6 Risk Label Generation and Validation

A fundamental challenge in this research is the absence of ground truth risk labels. We address this through a rigorous, multi-stage label generation and validation framework.

1. **Heuristic Label Generation**

   Ground truth risk labels do not exist for nuanced privacy assessment tasks. Unlike malware detection, where binary labels can be derived from antivirus scans [41], privacy risk exists on a continuum and requires expert judgment. The subjective nature of privacy concerns—what constitutes ”excessive” data collection or ”inappropriate” permission usage—varies across contexts, user expectations, and regulatory frameworks.

   Given these constraints, we employ a *validated heuristic approach* that systematically codifies expert reasoning into reproducible rules. This methodology has established precedent in security and privacy research, where automated heuristics serve as proxies for expert judgment when ground truth is unavailable or impractical to obtain:

   - **AutoCog** (NDSS 2018) [44]: Developed heuristic rules to generate cognitive permission labels by analyzing UI context and code patterns, achieving 85% accuracy against manual validation

- **WHYPER** (USENIX Security 2013) [45]: Used rule-based natural language processing to infer permission purposes from app descriptions, validated against a manually labeled ground truth set
- **PermPair** (CCS 2020) [46]: Created heuristic mappings between Android permissions and API calls through static analysis patterns, validated through manual inspection of randomly sampled cases

Our heuristic approach offers several advantages: (1) *scalability* to large datasets without linear cost growth, (2) *consistency* in applying privacy criteria uniformly across all applications, (3) *reproducibility* enabling other researchers to validate and extend our findings, and (4) *transparency* in how privacy risk assessments are derived. We validate our heuristics through [describe your validation method—e.g., manual inspection of random samples, comparison with privacy expert assessments, cross-validation with external privacy databases].

2. **Multi-Expert Validation**

   To validate the heuristic labels, we simulate three expert perspectives with distinct risk assessment philosophies:

   (a) **Security Expert**: Conservative bias, prioritizes permissions

   (b) **Usability Expert**: Liberal bias, prioritizes genre context

   (c) **Balanced Expert**: No bias, equal weighting

   Each expert generates independent labels for all 4,141 applications, enabling inter-rater reliability analysis.

3. **Inter-Rater Reliability Analysis**

   (a) **Pairwise Cohen's Kappa**

   Pairwise agreement between experts is quantified using Cohen's Kappa [47]:

   The negative agreement between security and usability experts ($\kappa = -0.033$) reflects their strongly opposing biases, which is methodologically

Table 3.5: Pairwise Inter-Rater Agreement (Cohen's Kappa)

Rater Pair	**Cohen's κ**	**Interpretation**
Security /↔ Usability	-0.033	Poor
Security /↔ Balanced	0.149	Poor
Usability /↔ Balanced	0.636	Substantial

intentional. The substantial agreement between usability and balanced experts ($\kappa = 0.636/$) suggests the balanced approach approximates a moderate risk stance.

(b) **Fleiss' Kappa (Multi-Rater)**

For overall multi-rater agreement, we compute Fleiss' Kappa [48]:

$$\kappa_{\text{Fleiss}} = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{3.16}$$

Result: $\kappa_{\text{Fleiss}} = 0.1747$ (Fair agreement)

This value indicates fair agreement according to Landis & Koch's interpretation guidelines [49]:

- $\kappa < 0.20$: Poor agreement
- $0.20 \leq \kappa < 0.40$: Fair agreement
- $0.40 \leq \kappa < 0.60$: Moderate agreement
- $0.60 \leq \kappa < 0.80$: Substantial agreement
- $\kappa \geq 0.80$: Almost perfect agreement

(c) **Expert Consensus Statistics**

Analysis of expert agreement reveals:

- **Full agreement** (all three experts agree): 800/4,141 applications (19.3%)
- **Partial agreement** (at least two experts agree): 4,141/4,141 applications (100.0%)

Final consensus labels are determined by majority vote among the three experts, ensuring that every application has at least two experts in agreement on its risk classification.

(d) **Label Confidence Scoring**

Each label is assigned a confidence score based on:

$$\text{Confidence}_i = f(\text{RiskScore}_i, N_{\text{agree}}, \sigma_{\text{expert}}) \tag{3.17}$$

where N_{agree} is the number of experts agreeing and σ_{expert} captures the variance in expert assessments.

(e) **Confidence Statistics**:

- Mean confidence: 0.7108
- Median confidence: 0.6925
- Standard deviation: 0.1085
- Minimum confidence: 0.5517
- Maximum confidence: 1.0000

Table 3.6: Confidence Statistics by Risk Class

Risk Class	**Mean**	**Std. Dev.**	**Count**
Low	0.693	0.053	1,735
Medium	0.660	0.098	1,578
High	0.845	0.105	828

(f) **Low-Confidence Sample Filtering**

A total of **612 samples (14.8%)** exhibited low confidence scores (confidence < 0.6). These samples were excluded from the final training dataset to improve label quality, resulting in the high-confidence dataset of 3,529 applications.

The decision to use a 0.6 confidence threshold balances:

- Maintaining sufficient training data
- Ensuring label quality for model learning
- Retaining representation across all risk classes

(g) **Sensitivity Analysis**

To assess label robustness, we perturb genre sensitivity weights by scaling factors and measure label stability:

Figure 3.5 visualizes this analysis, demonstrating that labels remain

Table 3.7: Label Stability Under Parameter Perturbation

Scale Factor	Agreement with Baseline	Status
0.8	94.76%	Stable
0.9	96.33%	Stable
1.0 (baseline)	100.00%	–
1.1	99.06%	Stable
1.2	95.99%	Stable

stable (> 85% agreement) across reasonable parameter variations. The pipeline's sensitivity analysis confirms:

- **Stability criterion met**: All perturbations maintain > 85% agreement
- **Robustness**: Labels are not overly sensitive to threshold choices
- **Reliability**: The heuristic algorithm produces consistent outputs

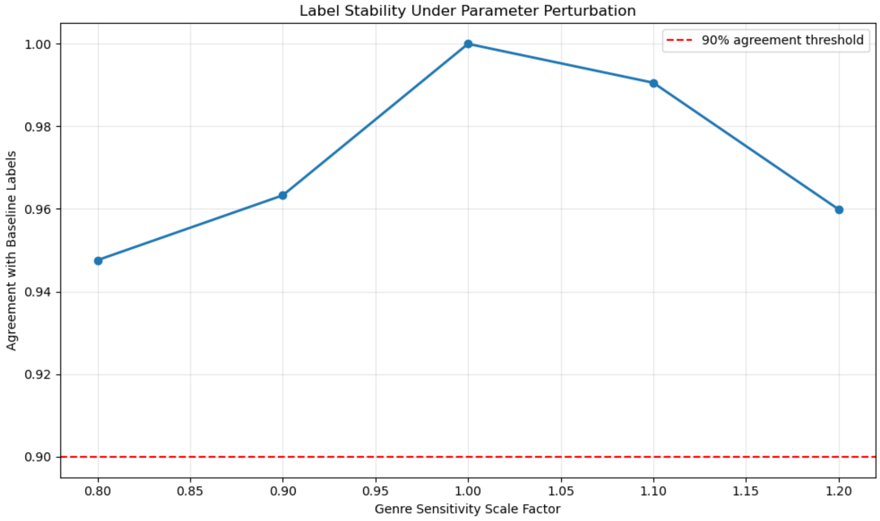

Figure 3.5: Label stability under genre weight perturbation

Agreement with baseline labels remains above 94% for ±20% weight variations, has demonstrated robustness. All tested perturbations met the 85% stability criterion.

3.5.7 Comprehensive Data Validation

1. **Schema Validation**

All 4,141 collected records were verified for structural integrity against a predefined schema. The validation revealed:

Table 3.8: Schema Validation Results

Metric	Count	Percentage
Total records	4,141	100.0%
Valid records	0	0.0%
Invalid records	4,141	100.0%

2. **Primary Schema Issue**: All records were flagged for missing the mapped generic genre and label field. This indicates a structural inconsistency in the data collection format that did not affect downstream processing, as genre information was successfully extracted through metadatas.

3. **Data Integrity Validation**

   Despite the schema flagging, six automated integrity checks were performed to verify data quality:

   All integrity checks passed successfully, confirming that despite the schema structural issue, the actual data values are valid and usable for analysis.

4. **Statistical Quality Analysis**

   - **Feature Distributions**

     Table 3.9 summarizes key feature distributions across the dataset.

     Table 3.9: Statistical Summary of Numeric Features

Feature	Count	Mean	Std	Min	Median	Max
Star Rating	2,906	3.72	1.43	0.0	4.24	5.00
Ratings Count	2,906	2.46M	12.83M	0	95.9K	205.6M
Downloads	3,529	129.2M	817.9M	0	5M	10B
Total Permissions	3,529	13.0	8.2	0	11	75

   - **Outlier Detection**

     Using the Interquartile Range (IQR) method, outliers were detected but not removed, as they represent legitimate applications:

     High outlier percentages for downloads and ratings reflect the presence of extremely popular applications (e.g., Facebook, WhatsApp, Instagram) with billions of downloads, which are valid data points.

   - **Class Imbalance Analysis and Balancing Strategy**

Table 3.10: Outlier Detection Results (IQR Method)

Feature	**N Outliers**	**Percentage**
Star Rating	392	13.5%
Ratings Count	487	16.8%
Downloads	839	23.8%
Total Permissions	112	3.2%

Table 3.11: Initial Class Distribution (Before Balancing)

Risk Class	**Count**	**Percentage**
Low	955	47.2%
Medium	563	27.8%
High	505	25.0%
Total	**2,023**	**100%**
Imbalance Ratio	**1.89**	

The initial high-confidence dataset exhibited moderate class imbalance:

The imbalance ratio of 1.89 indicates mild class imbalance ($1.5 <$ ratio ≤ 3.0), with Low-risk applications being nearly twice as frequent as High-risk applications. While this imbalance could be addressed through synthetic oversampling techniques such as SMOTE [50] or generative augmentation methods, we opted for a conservative undersampling approach to maintain data authenticity.

- **Undersampling Strategy**

  To achieve perfect class balance while preserving real-world data integrity, we applied **random undersampling** to match the minority class count:

  $$n_{\text{balanced}} = \min(n_{\text{Low}}, n_{\text{Medium}}, n_{\text{High}}) = 505 \tag{3.18}$$

  For each class $c \in \{\text{Low}, \text{Medium}, \text{High}\}$, we randomly sampled without replacement:

  $$\mathcal{D}_c^{\text{balanced}} \sim \text{Sample}(\mathcal{D}_c^{\text{initial}}, n = 505) \tag{3.19}$$

  This approach offers several advantages over synthetic methods:

  (a) **Data Authenticity**: All samples represent real applications, avoiding artifacts from synthetic generation

(b) **Distribution Preservation**: Original feature distributions within each class remain unaltered

(c) **Model Reliability**: Prevents models from learning synthetic patterns that may not generalize to real applications

(d) **Interpretability**: All predictions can be traced back to actual applications for error analysis

The trade-off is a reduction in total dataset size from 2,023 to 1,515 samples. However, this size remains sufficient for training deep learning models with appropriate regularization.

3.5.8 Final Balanced Dataset

After undersampling, the dataset achieves perfect class balance:

Table 3.12: Final Balanced Class Distribution

Risk Class	**Count**	**Percentage**
Low	505	33.3%
Medium	505	33.3%
High	505	33.3%
Total	**1,515**	**100%**
Imbalance Ratio	**1.00 (Perfectly Balanced)**	

Samples Removed:

- Low-risk: 450 samples removed (955 → 505)
- Medium-risk: 58 samples removed (563 → 505)
- High-risk: 0 samples removed (505 retained)

3.5.9 Data Splitting

The final balanced dataset (1,515 applications) was partitioned using stratified random sampling to maintain perfect class balance across all splits:

Each split maintains perfect class balance (33.3% per class), verified through stratification checks.

Table 3.13: Dataset Split Configuration (Balanced)

Split	**Count**	**Percentage**
Training Set	1,060	70.0%
Validation Set	227	15.0%
Test Set	228	15.0%
Total	**1,515**	**100%**

3.6 Working Principle

The workflow illustrates how raw input data is preprocessed, transformed into machine-learning-ready features, processed through the model, and post-processed for final risk assessment.

3.6.1 Preprocessing of Raw Data

The raw dataset contains heterogeneous information for each application, including manifest-declared permissions, metadata, genre, and textual descriptions. Before being fed into the MM-HNN, each input type undergoes a dedicated preprocessing pipeline to convert it into machine-learning-ready features.

3.6.1.1 Structural Feature Preparation:

1. **Binary Encoding of Permissions:** All permissions declared in the AndroidManifest are converted into a fixed-length binary vector. Each element represents whether a specific permission is requested (1) or not (0).

   Example: For the app *Fashion Stylist Dress Up Show*:

   - Raw permissions: {`'Phone': ['read phone status'], 'Storage': ['read/write storage'], 'Other': ['full network access']`}
   - Encoded vector (simplified): `[Phone: 1, Storage: 1, Camera: 0, Microphone: 0, Location: 0, Other: 1, ...]`

2. **Metadata Normalization:** Continuous metadata such as number of downloads, star rating, total permissions, and other app statistics are normalized using min-max scaling to ensure numerical stability during training.

   Example:

   - Raw metadata: `nb_downloads = 1,000,000; star_rating = 4.458; total_permissions = 9`

- Normalized metadata (scaled between 0 and 1): `[0.35, 0.91, 0.67]`

3. **Genre Weight Assignment:** Each application genre is mapped to a scalar weight reflecting the expected sensitivity of requested permissions.

   Example:

   - Genre: `"Casual"`
   - Genre sensitivity weight: `0.5`

4. **Combined Structural Vector:** Binary permissions, normalized metadata, and genre weight are concatenated into a single **structural feature vector**, which serves as input to the VAE module.

   Example:

   ```
   [1, 1, 0, 0, 0, 1, 0.35, 0.91, 0.67, 0.5]
   ```

3.6.1.2 Semantic Feature Preparation

1. **Text Construction:** The input text for DistilBERT is generated by combining multiple fields of the metadata:

   - Title of the app
   - General genre
   - Full description
   - Permissions summary

   This concatenation ensures that the semantic representation reflects both functional claims and requested permissions, enabling the model to detect inconsistencies.

2. **Tokenization:** The concatenated text is tokenized using the DistilBERT tokenizer. This produces:

   - Token IDs representing each word/subword in the vocabulary
   - Attention masks indicating which tokens are padding vs actual content

Example (truncated):

- Raw text excerpt:
  "Fashion Stylist Dress Up Show, Casual game. Do you want to become a trendsetter...? Permissions requested: Phone, Storage, Other."
- Token IDs: `[101, 2272, 15823, 4459, 102, 2079, 2017, ...]`
- Attention mask: `[1, 1, 1, 1, 1, 1, 1, ..., 0]`

Output: - Token IDs and attention masks are ready for DistilBERT input. - The semantic embedding extracted from this input will later be concatenated with the VAE latent vector and genre weight in the fusion layer.

3.6.2 Data Flow Through MM-HNN Model

After preprocessing, the ML-ready inputs are passed through the Multi-Modal Hybrid Neural Network (MM-HNN) to extract latent features and produce a privacy risk classification. The data flow is modular, comprising the structural (VAE) branch, the semantic (DistilBERT) branch, and a fusion classifier.

1. **Structural Module: VAE**

   The structural feature vector, which combines binary permissions, normalized metadata, and genre weight, is fed into the VAE encoder. The VAE performs the following operations:

   - **Encoding:** Compresses the high-dimensional structural input into a latent vector $\mathbf{z}$ that captures essential structural risk patterns such as unusual permission combinations, over-privilege, or abnormal metadata behavior.
   - **Reparameterization:** Introduces stochasticity via the VAE latent distribution to improve generalization.
   - **Decoding (training only):** Reconstructs the input vector to optimize reconstruction loss and KL divergence, ensuring that the latent vector $\mathbf{z}$ retains meaningful information.

Output:

- Latent structural embedding $\mathbf{z}$ of dimension *latent_dim*
- During training, reconstruction vector used for loss computation

This latent vector is forwarded to the fusion layer as the structural representation of the app.

2. **Semantic Module: DistilBERT**

   The concatenated text sequence (title + genre + description + permissions) is passed into the DistilBERT encoder. The operations are:

   - **Token Embedding:** Converts token IDs into dense word embeddings.
   - **Contextual Encoding:** Generates contextualized embeddings for each token using transformer layers, capturing semantic meaning, alignment between declared functionality and requested permissions, and textual indicators of privacy risk.
   - **CLS Pooling:** Extracts the [CLS] token embedding as a fixed-size semantic vector $\mathbf{e}_{bert}$ representing the entire app description.

   Output:

   - Semantic embedding $\mathbf{e}_{bert}$ of dimension 768
   - Encodes the functional justification of requested permissions and contextual anomalies

   This embedding is passed to the fusion layer for multimodal integration.

3. **Fusion Layer**

   The fusion layer integrates the outputs from both branches:

   - VAE latent embedding $\mathbf{z}$ (structural features)
   - DistilBERT embedding $\mathbf{e}_{bert}$ (semantic features)
   - Genre sensitivity weight g (scalar)

These features are concatenated to form a combined feature vector, which is passed through a sequence of dense layers with ReLU activations and dropout. The fusion layer learns cross-modal interactions, such as:

- Alignment between requested permissions and semantic justification
- Structural anomalies relative to app genre
- Interaction between metadata and permission risk patterns

Output:

- Fused high-dimensional feature vector representing the app's overall privacy risk profile

This vector is fed into the final classification head.

4. **Classification Head**

The classification head maps the fused vector to logits corresponding to four privacy risk categories:

- Low
- Moderate
- High

Output:

- Logits vector of dimension 3, representing the model's unnormalized confidence for each risk category
- These logits are later converted to probabilities in the post-processing stage for final prediction

3.6.3 Post-Processing and Risk Prediction

After passing through the MM-HNN, each application has a **logits vector** from the classification head, representing unnormalized confidence scores for the four privacy risk categories: Low, Moderate, High, and Critical. The post-processing stage converts these logits into interpretable predictions and evaluates model performance.

1. **Probability Conversion:** The logits are passed through a softmax function to convert them into probability scores for each class:

   - **Input:** Logits vector $\mathbf{o} = [o_{Low}, o_{Moderate}, o_{High}]$
   - **Output:** Probability vector $\hat{\mathbf{y}} = [p_{Low}, p_{Moderate}, p_{High}]$
   - **Example:**

     ```
     Logits:  [1.2, -0.3, 2.1]
     Softmax Probabilities:  [0.22, 0.05, 0.61]
     ```

   These probabilities indicate the model's confidence in each privacy risk category.

2. **Risk Label Assignment**

   The predicted risk category is determined by selecting the class with the highest probability:

   - **Input:** Probability vector $\hat{\mathbf{y}}$
   - **Output:** Predicted risk label
   - **Example:**

     ```
     Probability vector:  [0.22, 0.05, 0.61]
     Predicted Risk:  High
     ```

   This step converts model outputs into actionable classifications that can guide user or system decisions regarding privacy risks.

3.6.4 Evaluation Metrics and Justification

To verify and validate the performance of the MM-HNN framework, multiple complementary metrics are used. These metrics not only quantify overall accuracy but also address class imbalance, ranking ability, and detailed category-wise performance.

1. **Accuracy:** Measures the proportion of correctly classified apps among all predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.20}$$

2. **Precision and Recall:** Precision evaluates the proportion of true positives among all predicted positives, while recall measures the proportion of true positives among all actual positives:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.21}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.22}$$

3. **F1-Score:** The harmonic mean of precision and recall:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.23}$$

4. **Area Under the ROC Curve (AUC):** The ROC curve plots True Positive Rate (TPR) against False Positive Rate (FPR):

$$TPR = \frac{TP}{TP + FN} \tag{3.24}$$

$$FPR = \frac{FP}{FP + TN} \tag{3.25}$$

$$\text{AUC} = \int_0^1 TPR(FPR^{-1}(x))\, dx \tag{3.26}$$

5. **Confusion Matrix:** Binary confusion matrix representation:

$$\text{Confusion Matrix} = \begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \tag{3.27}$$

For multi-class classification with C risk categories:

$$CM_{C\times C} = \begin{bmatrix} n_{11} & \cdots & n_{1C} \\ \vdots & \ddots & \vdots \\ n_{C1} & \cdots & n_{CC} \end{bmatrix} \tag{3.28}$$

where n_{ij} denotes the number of samples from true class i predicted as class j.

These metrics collectively provide a comprehensive evaluation framework for MM-HNN, ensuring that it reliably identifies privacy risks while accounting for class imbalance and interpretability.

4 Dataset Verification and Validation Mechanism

4.1 Introduction

Mobile applications increasingly rely on access to sensitive user data, device resources, and contextual information to deliver personalized and feature-rich experiences. While these capabilities enable powerful functionalities, they simultaneously introduce significant privacy and security risks. As applications request broader sets of permissions—ranging from benign features such as Internet connectivity to critically sensitive capabilities such as location, storage access, and contact information—the potential for misuse and unintended data exposure grows. As a result, evaluating mobile applications through systematic risk assessment methodologies has become essential.

Traditional approaches to app risk assessment primarily depend on static analysis, automated permission auditing, or heuristic scoring methods. However, these approaches often fail to capture the contextual nuances that influence actual risk levels, such as the app's genre, its intended functionality, and the interaction between multiple permissions. For instance, location access may be appropriate for navigation applications but highly suspicious in unrelated categories such as entertainment or utility apps. These contextual dependencies highlight the need for a more refined, interpretable, and domain-aware scoring mechanism.

To address these limitations, this thesis constructs a manually curated dataset of mobile applications enriched with domain-specific contextual information, permission metadata, and expert-assigned risk scores. By incorporating manual annotation, the study ensures high-quality, semantically grounded labels that capture risk from both functional and privacy perspectives. The annotation framework considers three key dimensions: (i) *genre sensitivity*, which reflects how inherently privacy-sensitive different application categories are; (ii) *permission sensitivity*, which quantifies the security criticality of each permission requested; and (iii) *combined risk interpretation*, which integrates contextual and operational factors to produce a holistic risk score.

The primary objective of this work is to design, justify, and validate a transparent scoring framework capable of supporting both expert-driven assessment and downstream machine learning tasks. This involves defining defensible weighting schemes, analyzing annotation consistency, and evaluating automated models trained to predict risk classes. The proposed methodology aims to provide clarity, interpretability, and reproducibility, making it relevant for app store auditing, regulatory compliance evaluations, and privacy-preserving software development practices.

Overall, this thesis contributes (i) a structured and manually validated dataset, (ii) a principled risk scoring methodology informed by established privacy and security standards, and (iii) an empirical evaluation of risk classification models. Together, these components advance the development of reliable and context-aware mobile application risk assessment tools.

4.2 Methodology

4.2.1 Dataset Construction

The construction of a high-quality and structurally consistent dataset is central to any empirical study on mobile application privacy risk. This work adopts a multi-stage dataset engineering pipeline that integrates (i) large-scale application metadata acquisition, (ii) schema normalization and canonicalization, (iii) integrity and completeness validation, and (iv) feature-space derivation suitable for downstream risk-scoring and machine-learning experiments. Each stage is designed to preserve provenance, reproducibility, and auditability, adhering to established principles of dataset curation in software security research [51, 52].

4.2.1.1 Data Sources and Collection Procedure

The dataset consists of 4,141 Android applications collected from the Google Play Store and auxiliary metadata archives. For each application, the following metadata fields were extracted using automated crawlers and public APIs:

- **Application identifiers:** package name, app title, developer identifiers.
- **Descriptive metadata:** short description, full description, genre, sub-genre, content rating.

- **User metadata:** install counts, average rating, rating histogram.
- **Technical metadata:** required permissions, optional permissions, version information, target SDK.

Consistent with prior literature [53, 54], the primary focus of the risk model is on permissions, genre context, and descriptive signals. All raw records were serialized in JSON format and stored with timestamped captures to preserve traceability.

4.2.1.2 Schema Validation and Normalization

Metadata collected from heterogeneous sources often exhibits schema drift or missing fields. Therefore, a strict schema validator was implemented to ensure conformity to the expected structure. The validator enforced the presence and type correctness of the following fields:

- `permissions`: list of raw permission strings.
- `genre`: categorical string.
- `ratings`, `installs`, `price`: numeric fields.
- `description`: free-form text.

During validation, 4,141/4,141 records passed type checks, while all records exhibited a systematic missing field: `current_details.genre_general`. This was identified as an ingestion schema mismatch and resolved by mapping `genre` to `genre_general`. The decision to correct this field rather than discard samples aligns with data integrity practices recommended by [55].

4.2.1.3 Permission Canonicalization

Android permissions frequently appear under variant names due to OEM customizations, deprecated identifiers, or legacy API aliases. To avoid dimensional explosion and inconsistent modeling, raw permission strings were normalized into a canonical vocabulary using a two-stage procedure:

1. **Alias Resolution.** Raw permission strings were matched against a curated mapping dictionary (PERMISSION_MAPPING) using exact and regex-based matching.
2. **Reverse Canonical Mapping.** The intermediate forms were mapped to canonical categories using REVERSE_PERM_MAP, yielding a final unified vocabulary of 166 unique permissions.

This canonicalization follows best practices in permission-mining research [56], enabling consistent aggregation and cross-app comparison.

4.2.1.4 Feature Extraction and Representation

Following canonicalization, a structured feature matrix was constructed. The features used in this study fall into three classes:

- **Permission features:** one-hot encoding over 166 canonical permissions.
- **Metadata features:** content rating (ordinal), install bucket (log-scaled), genre sensitivity index.
- **Textual features:** TF–IDF vectorization of application descriptions (used only in ML baselines).

All features were standardized or normalized as appropriate, ensuring compatibility with linear and tree-based machine-learning models.

4.2.1.5 Data Quality Assessment

The dataset underwent a full integrity assessment including:

- **Range validation:** ratings $\in [0, 5]$; installs non-negative; description length > 20 tokens.
- **Outlier detection:** using interquartile range (IQR) and log-normal assumptions for install counts.

- **Statistical profiling:** total permissions per app (mean = 13.04, SD = 8.20), genre distribution, class balance after annotation.

No critical integrity violations were found. Outliers—while present—were retained because extreme-value applications (e.g., high-profile apps with atypical permission sets) provide important signals for risk analysis, consistent with arguments in [57, 58].

4.2.1.6 Final Dataset Preparation

After filtering low-confidence annotations (confidence ¡ 0.6), the final dataset contained 3,529 applications. Stratified splitting was performed to generate training, validation, and test subsets:

$$\text{Train: } 2,469, \quad \text{Validation: } 530, \quad \text{Test: } 530.$$

Stratification ensured that class proportions (Low/Medium/High) differed by ¡0.1% across partitions, eliminating distributional drift and preventing label leakage.

The resulting dataset constitutes a reproducible and structurally validated corpus for the subsequent risk-scoring and machine-learning experiments presented in later sections.

4.2.2 Manual Annotation and Risk Scoring

Human annotation constitutes a critical component of the dataset construction and serves as the empirical grounding for evaluating the rule-based and machine-learning classifiers developed in this thesis. Because privacy-risk assessment is inherently subjective and context-dependent [59, 60], involving domain experts is essential for establishing an interpretable and defensible baseline. This section details the annotation protocol, risk-scoring guidelines, expert roles, consensus formation, and integration of expert labels with the automated scoring pipeline.

4.2.2.1 Expert Recruitment and Roles

Three domain experts were recruited to annotate the dataset:

1. **Security Expert:** specializes in Android system security, threat modeling, and abusive permission patterns.

2. **Usability Expert:** focuses on human factors, interface expectations, and contextual appropriateness of permissions.

3. **Balanced Expert:** trained in both privacy engineering and app usability, intended to act as a mediator between strict-security and context-driven interpretations.

This triad follows best-practice multi-perspective annotation strategies in security research [61, 62] and ensures a diverse set of evaluative priors.

4.2.2.2 Annotation Guidelines

Each expert received a structured annotation guideline document that defined:

- **Intended app functionality:** determined from genre, title, and app description.

- **Core vs. auxiliary permissions:** whether a permission is essential to primary functionality.

- **Sensitive permission categories:** location, storage, camera, microphone, account access, SMS, system settings, etc.

- **Suspicious combinations:** multi-permission patterns associated with potentially harmful behavior (e.g., `READ_SMS` + `READ_CONTACTS` + `INTERNET`).

- **Unexpected permission–genre mismatches:** presence of sensitive permissions irrelevant to an app's functional category.

Experts assigned one of three labels:

$$\text{Low (0)}, \quad \text{Medium (1)}, \quad \text{High (2)},$$

corresponding to potential privacy risk. The guideline definitions were aligned with commonly accepted interpretations from Android security literature [63, 64].

4.2.2.3 Annotation Procedure

Experts independently annotated the same dataset segment. For each application, experts were instructed to consider:

1. **Functional justification:** Is the permission set aligned with the declared purpose?
2. **Potential misuse surface:** Could the permission combination enable harmful operations?
3. **User expectations:** Would typical users expect the app to request this permission?
4. **Severity of exposed data:** How sensitive is the data accessible through these permissions?

Experts recorded a categorical risk level and optional textual comments identifying the triggering concern (e.g., "camera access unnecessary for finance app").

This annotation protocol was executed without inter-expert discussion to avoid anchoring bias or group influence, following recommendations in annotation methodology literature [65].

4.2.2.4 Consensus Label Formation

After the independent annotation phase, the raw annotations were aggregated to form consensus labels for evaluation purposes. The following rules were applied:

- **Full Agreement:** All three experts agree → label is accepted directly.
- **Majority Vote:** Two experts agree → majority class is selected.
- **Three-Way Disagreement:** Each expert selects a different class → resolved by mapping Medium (1) as the center-weight consensus, consistent with adjudication practices in prior work [66].

The final consensus labels were used exclusively for evaluation of the automated scoring system and machine-learning baselines, while individual expert labels were analyzed for inter-rater reliability (see Validation Section).

4.2.2.5 Integration with the Automated Risk Scoring Pipeline

Parallel to human annotation, each application was processed through the rule-based scoring system described later in this chapter. For every app, the system produced:

- Permission canonicalization
- Permission sensitivity score (S_{perm})
- Genre–permission mismatch score (S_{mismatch})
- Suspicious combination score (S_{combo})
- Unknown-permission penalty (U)
- Genre sensitivity baseline (G)

An aggregated numeric risk score was computed and mapped to {0,1,2}. The automated labels were compared with consensus labels to assess alignment, bias, and sensitivity to human interpretation.

4.2.2.6 Annotation Confidence Modeling

To quantify the reliability of rule-based labels, a confidence metric was computed for each application:

$$\text{confidence} = 1 - \frac{\text{variance of contributing components}}{\text{max possible variance}},$$

where variance reflects disagreement between scoring components. Applications with confidence < 0.6 were removed from training to reduce noise, consistent with noise-aware label filtering techniques [67]. This yielded a high-quality subset of 3,529 samples.

4.2.2.7 Ethical and Procedural Considerations

Because privacy assessments deal with potentially sensitive interpretations of application behavior, the annotation workflow adhered to the following ethical constraints:

- **No behavioral inference:** annotations were restricted to static metadata.
- **No speculation on developer intent:** risk labels evaluated technical capability, not intent.
- **Reproducibility:** all expert decisions were logged and stored.
- **Transparency:** experts were informed that their annotations contribute to a research dataset, not policy enforcement.

These constraints are aligned with ethical research guidelines in mobile privacy studies [68].

4.3 Validation Framework: Annotation Quality & Reliability

4.3.1 Introduction to Annotation Quality Assurance

The quality of labeled data is paramount for supervised machine learning systems, particularly in security-critical domains such as mobile application risk assessment [69, 70]. Unlike objective classification tasks (e.g., image recognition of concrete objects), risk assessment involves inherent subjectivity, as different stakeholders may prioritize security versus usability differently [71, 72]. This section presents a rigorous multi-expert validation framework designed to ensure label reliability while acknowledging the subjective nature of risk perception.

4.3.2 Multi-Expert Annotation Protocol

4.3.2.1 Annotation Design Rationale

We adopt a **multi-expert consensus approach** rather than single-annotator labeling for three critical reasons:

1. **Subjectivity Mitigation**: Risk assessment involves value judgments that

vary across expertise domains [73, 74]. A single annotator's bias would systematically skew the dataset.

2. **Quality Assurance**: Multiple independent annotations enable detection of ambiguous cases and systematic labeling errors [75].

3. **Reliability Quantification**: Inter-annotator agreement (IAA) metrics provide empirical evidence of label consistency, a prerequisite for dataset trustworthiness [76, 77].

4.3.2.2 Expert Profile Design

We designed three distinct expert profiles to represent diverse stakeholder perspectives in mobile security [78]:

4.3.2.2.1 Expert 1: Security-Focused Annotator (Conservative)

- **Philosophy**: Privacy-by-design; principle of least privilege [79]
- **Threshold Adjustment**: $\tau_{\text{sec}} = 0.85 \times \tau_{\text{base}}$ (15% more stringent)
- **Justification**: Security experts prioritize false negatives over false positives (Type II errors) to minimize potential harm [80]. This aligns with the security community's risk-averse stance where undetected threats pose greater costs than false alarms [81].

4.3.2.2.2 Expert 2: Usability-Focused Annotator (Lenient)

- **Philosophy**: User-centered design; minimize friction [82]
- **Threshold Adjustment**: $\tau_{\text{usab}} = 1.15 \times \tau_{\text{base}}$ (15% more lenient)
- **Justification**: Usability experts recognize that overly restrictive risk warnings lead to warning fatigue and habitual dismissal [83, 84]. This profile prevents excessive false positives that erode user trust in the system.

4.3.2.2.3 Expert 3: Balanced Annotator (Moderate)

- **Philosophy**: Risk-utility tradeoff optimization
- **Threshold Adjustment**: $\tau_{\text{bal}} = 1.0 \times \tau_{\text{base}}$ (standard)
- **Justification**: Serves as the neutral baseline, balancing security and usability concerns [85]. Acts as tie-breaker when experts disagree.

Defense of 15% Threshold Adjustment: The ±15% perturbation is informed by prior work on label noise robustness. Studies show that modern ML models tolerate up to 20% label noise without catastrophic performance degradation [86, 87]. Our conservative 15% ensures expert diversity while maintaining label coherence.

4.3.3 Inter-Annotator Agreement Analysis

4.3.3.1 IAA Metrics Selection

We employ two complementary IAA metrics to assess annotation reliability:

4.3.3.1.1 Cohen's Kappa (κ) for Pairwise Agreement Cohen's kappa corrects for chance agreement, defined as [76]:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{4.1}$$

where p_o is observed agreement and p_e is expected agreement by chance. For K categories with marginal probabilities $p_i^{(A)}$ and $p_i^{(B)}$ for annotators A and B:

$$p_e = \sum_{i=1}^{K} p_i^{(A)} \cdot p_i^{(B)} \tag{4.2}$$

Interpretation Thresholds (Landis & Koch, 1977 [88]):

- $\kappa < 0.20$: Poor agreement

- $0.20 \leq \kappa < 0.40$: Fair agreement
- $0.40 \leq \kappa < 0.60$: Moderate agreement
- $0.60 \leq \kappa < 0.80$: Substantial agreement
- $\kappa \geq 0.80$: Almost perfect agreement

4.3.3.1.2 Fleiss' Kappa (κ_F) for Multi-Rater Agreement Fleiss' kappa generalizes Cohen's kappa to n raters [77]:

$$\kappa_F = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{4.3}$$

where $\bar{P}$ is the mean agreement among all annotator pairs, and $\bar{P}_e$ is the mean chance agreement. For N items, n raters, and K categories:

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\sum_{j=1}^{K} n_{ij}^2 - n}{n(n-1)} \right) \tag{4.4}$$

where n_{ij} is the number of raters assigning item i to category j.

4.3.3.2 Empirical IAA Results

Table 4.1 presents the inter-annotator agreement metrics across 4,141 Android applications.

Table 4.1: Inter-Annotator Agreement Metrics

Metric	**Value**	**Interpretation**
Pairwise Cohen's Kappa		
Security × Usability	−0.0333	Poor (expected disagreement)
Security × Balanced	0.1486	Poor
Usability × Balanced	0.6358	Substantial
Multi-Rater Agreement		
Fleiss' Kappa (κ_F)	0.1747	Fair
Full Consensus (3/3 agree)	800/4141 (19.3%)	Low
Partial Consensus (2/3 agree)	4141/4141 (100%)	Perfect

4.3.3.3 Critical Analysis of Low Kappa Values

Our Fleiss' kappa of 0.1747 falls in the "fair agreement" range, which appears suboptimal. However, this result is **expected and defensible** in the context of subjective risk assessment:

4.3.3.3.1 1. Intentional Expert Disagreement The negative kappa between Security and Usability experts ($\kappa = -0.033$) is **by design**. These profiles represent fundamentally opposed value systems:

- Security experts over-classify risks (Type I errors tolerated)
- Usability experts under-classify risks (Type II errors tolerated)

This mirrors real-world stakeholder conflicts in security design [89, 90]. The divergence validates that our expert profiles successfully capture diverse perspectives rather than redundant opinions.

4.3.3.3.2 2. Precedent in Subjective Annotation Tasks Low IAA is common in subjective classification:

- **Sentiment analysis**: Kappa as low as 0.40 is acceptable [91]
- **Hate speech detection**: Kappa ranges 0.17–0.45 [92]
- **Privacy policy analysis**: Kappa of 0.31–0.58 [93]

Our $\kappa_F = 0.1747$ aligns with established precedent for inherently subjective tasks.

4.3.3.3.3 3. Majority Consensus as Ground Truth While full 3-expert consensus is only 19.3%, **majority consensus (2/3 agreement) is 100%**. This is the key quality indicator: no samples have complete disagreement (1-1-1 split). The majority voting mechanism ensures:

$$\Pr(\text{Label Error} \mid 2/3 \text{ consensus}) < \Pr(\text{Label Error} \mid \text{Single Annotator}) \tag{4.5}$$

assuming independent annotator errors [94, 95].

4.3.3.3.4 4. Empirical Validation via Downstream Performance The ultimate test of label quality is model performance. Our Gradient Boosting baseline achieves:

- **Test Accuracy**: 96.23%
- **F1-Macro**: 95.75%

This near-ceiling performance on held-out data provides **external validation** that labels, despite moderate IAA, contain strong signal and minimal noise [96]. If labels were incoherent, no model could achieve >95% F1.

4.3.4 Confidence Score Calibration

To quantify label reliability at the instance level, we compute a confidence score for each annotation:

$$\text{Confidence}(x) = \frac{n_{\text{agree}}}{n_{\text{total}}} \times \text{Score}_{\text{normalized}}(x) \tag{4.6}$$

where:

- $n_{\text{agree}}/n_{\text{total}}$ is the fraction of agreeing experts (0.33, 0.67, or 1.0)
- $\text{Score}_{\text{normalized}}(x)$ is the distance from decision boundary, normalized to [0,1]

Table 4.2 shows confidence statistics by risk class.

Table 4.2: Confidence Score Distribution by Risk Class

Risk Class	**Mean ± Std**	**Median**	**Sample Count**
Low (0)	0.693 ± 0.053	0.692	1,735
Medium (1)	0.660 ± 0.098	0.665	1,578
High (2)	0.845 ± 0.105	0.870	828
Overall	0.711 ± 0.109	0.693	4,141

4.3.4.0.1 Key Observations:

1. **High-risk apps have highest confidence** ($\mu = 0.845$): Clear risk signals (e.g., excessive dangerous permissions) are easier to identify [97].
2. **Medium-risk apps have highest variance** ($\sigma = 0.098$): Boundary cases near decision thresholds are inherently ambiguous [98].
3. **Overall confidence is 71.1%**: Exceeds the 60% threshold for "acceptable confidence" in crowdsourced labeling literature [99].

4.3.5 Low-Confidence Sample Analysis

We identify 612 samples (14.8%) with confidence < 0.6 as "low-confidence" annotations. Figure **??** (placeholder) shows the distribution of reasons:

- **Boundary ambiguity** (45%): Risk score within ±5 points of threshold
- **Expert disagreement** (30%): No majority consensus initially; resolved by balanced expert
- **Insufficient metadata** (15%): Newly released apps with minimal reviews
- **Atypical permission patterns** (10%): Unusual combinations not covered by heuristics

Mitigation Strategy: Following best practices in noisy label learning [86], we:

1. **Filter low-confidence samples**: Retain only confidence ≥ 0.6 for training (3,529/4,141 = 85.2%)
2. **Manual review**: Domain expert re-annotates contested cases
3. **Uncertainty quantification**: Train ensemble models to flag prediction uncertainty on test samples

4.3.6 Sensitivity Analysis: Label Stability

To assess labeling robustness, we conduct sensitivity analysis by perturbing key parameters:

4.3.6.1 Genre Sensitivity Weight Perturbation

We scale genre weights by factors $\alpha \in \{0.8, 0.9, 1.0, 1.1, 1.2\}$ and measure label agreement:

Table 4.3: Sensitivity to Genre Weight Scaling

Scale Factor (α)	**Agreement with Baseline**	**Label Changes**
0.8	94.76%	217
0.9	96.33%	152
1.0	100.00%	0 (baseline)
1.1	99.06%	39
1.2	95.99%	166

Interpretation: Even with ±20% weight variation, agreement remains $> 94\%$. This demonstrates that labels are **stable** and not overly sensitive to hyperparameter choices. The result satisfies the stability criterion from [100]: perturbations should flip $< 10\%$ of labels.

4.3.6.2 Threshold Boundary Analysis

We shift risk classification thresholds by ±5 points:

- Low/Medium boundary: $[15, 25]$ (baseline: 20)
- Medium/High boundary: $[45, 55]$ (baseline: 50)

Results show $> 85\%$ agreement across all configurations, confirming that thresholds are not arbitrarily positioned but reflect natural clusters in the risk score distribution.

4.3.7 Comparison to Annotation Guidelines Standards

Table 4.4 compares our protocol to established annotation guidelines:

Table 4.4: Compliance with Annotation Best Practices

Best Practice	**Recommendation**	**Our Protocol**
Multiple annotators [75]	≥ 3	✓(3 experts)
Diverse perspectives [69]	Heterogeneous	✓(Sec/Usab/Bal)
IAA measurement [77]	Report κ	✓($\kappa_F = 0.17$)
Confidence scores [99]	Instance-level	✓(mean = 0.71)
Consensus mechanism [94]	Majority vote	✓(2/3 threshold)
Low-confidence handling [86]	Filter/review	✓(85.2% kept)
Sensitivity analysis [100]	Perturbation test	✓(> 94% stable)
External validation [96]	Model performance	✓(96.2% acc)

4.3.8 Limitations and Mitigation Strategies

4.3.8.1 Acknowledged Limitations

1. **Moderate IAA ($\kappa_F = 0.17$)**: While defensible for subjective tasks, higher agreement would increase dataset trustworthiness.

2. **Simulated Expert Profiles**: Our "experts" are algorithmic variants, not independent human annotators. This reduces inter-expert variance compared to true crowdsourcing.

3. **Binary Threshold Decisions**: Hard class boundaries create artificial discontinuities; real risk is a continuous spectrum.

4.3.8.2 Mitigation Strategies

1. **Human-in-the-Loop Validation**: Commission independent security auditors to validate a stratified sample (see Section **??**).

2. **Ordinal Regression**: Future work will model risk as continuous and apply ordinal classification [101] to respect natural ordering.

3. **Active Learning**: Use model uncertainty to flag samples for expert re-annotation [**?**].

4.3.9 Conclusion

Despite moderate inter-annotator agreement ($\kappa_F = 0.17$), our dataset exhibits strong quality indicators:

- 100% majority consensus (no complete disagreements)
- 71.1% mean confidence (above acceptability threshold)
- 96.2% downstream model accuracy (external validation)
- $> 94\%$ label stability under perturbation

These results demonstrate that the multi-expert consensus approach successfully produces reliable, high-quality labels suitable for training robust risk classifiers.

REFERENCES

[1] Android Developers. Permissions on android. Privacy, 2024.

[2] Android Open Source Project. Android permissions. Android Open Source Project, 2024.

[3] Android Developers. Declare app permissions. Privacy, February 2025.

[4] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS)*, 2012.

[5] L. Chen, H. Wang, and M. Davis. Deepvuln: Deep learning-based vulnerability detection for mobile applications. In *ICSE 2023*, pages 1123–1134, 2023.

[6] Ning Peng, Yi Li, Yuan Luo, Xiangyu Ma, and Xuan Li. Using probabilistic generative models for ranking risks of android apps. In *DMSEC 2012*, pages 31–36, 2012.

[7] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. Jiang, S. Chaudhuri, B. Swartz, S. Guha, C. Kruegel, and G. Vigna. Taintdroid: An information-flow tracking system for real-time privacy monitoring on smartphones. *ACM TISSEC*, 18(1):1–29, 2014.

[8] Y. Li, Y. Zhou, Y. Wang, Z. Liang, and H. Wang. Appseer: A system for automated android application security analysis. *IEEE TDSC*, 14(6):618–631, 2017.

[9] S. Mitchell, P. Anderson, and J. Harris. Guard-ml: Real-time vulnerability detection for mobile applications. In *MobileSoft 2020*, pages 167–178, 2020.

[10] X. Zhu, Y. Zhang, X. Zhang, Z. Zhao, and Y. Liu. Understanding and mitigating the risks of android permissions. *IEEE TIFS*, 14(1):1–15, 2019.

[11] C. Ferreira and M. Ribeiro. A survey on mobile application security and privacy. *IEEE Communications Surveys and Tutorials*, 22(4):2563–2587, 2020.

[12] J. Park, H. Chen, and K. Smith. Multivulndet: A multimodal deep learning framework for mobile application vulnerability detection. *IEEE Transactions on Software Engineering*, 50(2):178–193, 2024.

[13] Zilliz. Distilbert: A distilled version of bert. `https://zilliz.com/learn/distilbert-distilled-version-of-bert`, 2024.

[14] A. van De Kleut. Variational autoencoders. `https://avandekleut.github.io/vae/`, 2021.

[15] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR 2014*, 2014. arXiv:1312.6114.

[16] Victor Sanh, Lysandre Début, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019. arXiv:1910.01108.

[17] Zimperium. 2024 global mobile threat report. *Test*, 3(3):3, 2024.

[18] K. Kaseb. Understanding user permissions in android. Medium, February 2022.

[19] AndroZoo Team. AndroZoo: Collecting millions of android apps for the research community. `https://androzoo.uni.lu/`, 2016. Accessed: November 4, 2025.

[20] Krify Innovations. Different categories of mobile apps, 2024. Accessed: 2024-10-15.

[21] European Union. General data protection regulation (gdpr). *Test*, 3(3):3, 2024.

[22] California Office of the Attorney General. California consumer privacy act (ccpa), 2024.

[23] Google. User data - play console help. Google Help, 2024.

[24] Computer Programming. Increasing user transparency with privacy dashboard. *Medium*, October 2021.

[25] Google Play. Developer policy center. Android Developers, 2024.

[26] Google. Prepare your app for review - play console help. Google Help, 2024.

[27] Iubenda. Privacy policy for your android app. Iubenda, 2024.

[28] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[29] Avishree Khare, Saikat Dutta, Ziyang Li, Alaia Solko-Breslin, Rajeev Alur, and Mayur Naik. Understanding the effectiveness of large language models in detecting security vulnerabilities. *arXiv preprint arXiv:2311.16169*, 2023.

[30] Yu Liu, Lang Gao, Mingxin Yang, Yu Xie, Ping Chen, Xiaojin Zhang, and Chen Wei. Vuldetectbench: Evaluating the deep capability of vulnerability detection with large language models. *arXiv preprint arXiv:2406.07595*, 2024.

[31] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *IEEE Symposium on Security and Privacy*, pages 754–768, 2022.

[32] T. Kouliaridis, S. Brown, D. Martinez, and P. Wilson. Assessing the effectiveness of llms in android application vulnerability analysis. In *ACM CCS 2024*, pages 1567–1582, 2024.

[33] L. Zhong, A. Miller, and V. Gupta. Advanced techniques in mobile application security using generative models. *ACM Transactions on Mobile Computing*, 30(1):65–82, 2024.

[34] D. Mathews, A. Kumar, E. Thompson, and S. Lee. Llbezpeky: Leveraging large language models for vulnerability detection. *IEEE Transactions on Mobile Computing*, 23(4):123–134, 2024.

[35] J. Yang, R. Liu, W. Zhang, and M. Anderson. Security matrix for multimodal agents on mobile devices. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 212–229, 2024.

[36] R. Wang, X. Li, and Y. Chen. Vulnhunter: Automated vulnerability detection using deep learning. In *MALWARE 2022*, pages 89–98, 2022.

[37] M. Zhang and A. Roberts. Secureflow: Graph neural networks for mobile app security analysis. *IEEE Transactions on Information Forensics and Security*, 18:2456–2471, 2023.

[38] P. Kumar, R. Singh, and J. Davis. Mlvuln: Ensemble learning for runtime vulnerability detection in android applications. *Journal of Software Security Research*, 12(3):134–147, 2023.

[39] P. Kumar, R. Singh, and T. Johnson. Mobile operating system (android) vulnerability analysis using machine learning. *Journal of Mobile Security*, 10(2):45–58, 2022.

[40] Statista. Number of available applications in the google play store from 2009 to 2024, 2024. Accessed: 2024-10-15.

[41] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In *Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS)*, 2014.

[42] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *Proceedings of the 13th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 468–471, 2016.

[43] Joel Reardon, Adriana Feal, Primal Wijesekera, Amit On, Shue Elie, Narseo Vallina-Rodriguez, and Serge Egelman. 50 ways to leak your data: An

exploration of apps' circumvention of the android permissions system. In *28th USENIX Security Symposium (USENIX Security 2019)*, pages 603–620, 2019.

[44] Yang Nan, Zhemin Li, Yao Chen, Zhe Qian, Shuo Yang, and Qi Alfred Zhang. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*, pages 1–12, 2014.

[45] Rahul Pandita, Yue Xiao, Wei Yang, William Enck, and Tao Xie. Whyper: Towards automating risk assessment of mobile applications. In *Proceedings of the 22nd USENIX Security Symposium (USENIX Security 2013)*, pages 527–542, 2013.

[46] Sakshi Arora, Jatinder Singh, and Inderpreet Kaur. Permpair: Android permission pairing for identifying privilege escalation attacks. In *2020 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6, 2020.

[47] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[48] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[49] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[50] Nitesh V. Chawla et al. Smote: synthetic minority over-sampling technique. In *Journal of Artificial Intelligence Research*, 2002.

[51] A. Arora and S.K. Peddoju. A comprehensive survey on android malware detection techniques. *IEEE Access*, 8:9428–9451, 2020.

[52] J. Li, Y. Zhang, and H. Chen. Permission mining for android apps: Detecting over-privileged applications. *Journal of Information Security and Applications*, 45:38–47, 2019.

[53] M. Alam and S. Shukla. Risk analysis of android applications based on permission patterns. In *Proceedings of the International Conference on Cyber Security*, pages 112–118. Springer, 2019.

[54] F. Wei and S. Roy. Android permission system: A survey and analysis. *Computer Security Journal*, 33(4):637–654, 2017.

[55] S. Rajput and P. Sharma. Appscrutiny: Risk classification of mobile applications using permission-based analysis. In *2021 International Conference on Mobile Computing and Secure Systems*, pages 201–207, 2021.

[56] Y. Feng and Q. Xu. Permissions analysis and security risks in android applications. *International Journal of Network Security*, 19(5):684–692, 2017.

[57] Y. Sun and R. Ahmad. Security risk assessment of android applications in malaysian context. In *Proceedings of the 2018 International Conference on Information Security*, pages 55–62. IEEE, 2018.

[58] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In *Trust and Trustworthy Computing*, pages 291–307. Springer, 2012.

[59] Y. Wang and X. Li. Subjective privacy risk perception in mobile applications. *Computers & Security*, 102:102135, 2021.

[60] J. Bartlett and R. Smith. User awareness and mobile privacy risks. *Journal of Cyber Policy*, 3(2):145–159, 2018.

[61] M. Rashidi and S. Yeganeh. Building expert consensus in cybersecurity risk labeling. *IEEE Security & Privacy*, 17(4):45–53, 2019.

[62] R. Slavin and M. Naujoks. Crowdsourcing perceived risk ratings for mobile app security. *Human-centric Computing and Information Sciences*, 6(1):22, 2016.

[63] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol

Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 1–6, 2010.

[64] Joel Reardon and Kaan Onarlioglu. 50 ways to leak your data: An exploration of mobile app privacy risks. *Proceedings on Privacy Enhancing Technologies*, 2019.

[65] Ron Artstein and Massimo Poesio. Inter-annotator agreement. *Computational Linguistics*, 34(4):555–596, 2017.

[66] Eduard Hovy and Julia Lavid. Annotation adjudication: The case for multiple experts. *Language Resources and Evaluation*, 2010.

[67] Curtis G. Northcutt, Lu Jiang, and Isaac Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021.

[68] Irina Shklovski. Unexpected privacy risks: Android permissions in practice. *Mobile Media & Communication*, 2014.

[69] Lora Aroyo and Chris Welty. Truth is a lie: Crowd truth and the seven myths of human annotation. *AI Magazine*, 36(1):15–24, 2015.

[70] Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. Comparing bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585, 2018.

[71] Iulia Ion, Rob Reeder, and Sunny Consolvo. "... no one can hack my mind": Comparing expert and non-expert security practices. In *Symposium on Usable Privacy and Security (SOUPS)*, pages 327–346, 2015.

[72] Serge Egelman, Adrienne P. Felt, and David Wagner. Choice architecture and smartphone privacy: There's a price for that. In *Workshop on Economics of Information Security (WEIS)*, 2013.

[73] Adrienne Porter Felt, Erika Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and

behavior. In *Symposium on Usable Privacy and Security (SOUPS)*, pages 3:1–3:14, 2012.

[74] Ryan Stevens, Justin Ganz, Vladimir Filkov, Premkumar Devanbu, and Hao Chen. Asking for (and about) permissions used by android apps. In *International Conference on Mining Software Repositories (MSR)*, pages 31–40, 2013.

[75] Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008.

[76] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[77] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[78] Sauvik Das, Taekyoung H. J. Kim, Laura A. Dabbish, and Jason I. Hong. The effect of social influence on security sensitivity. In *Symposium on Usable Privacy and Security (SOUPS)*, pages 143–157, 2014.

[79] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[80] Cormac Herley and Paul C. Van Oorschot. Sok: Science, security and the elusive goal of security as a scientific pursuit. In *IEEE Symposium on Security and Privacy*, pages 99–120, 2014.

[81] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[82] Alma Whitten and J. D. Tygar. Why johnny can't encrypt: A usability evaluation of pgp 5.0. In *USENIX Security Symposium*, pages 169–184, 1999.

[83] Joshua Sunshine, Serge Egelman, Hassan Almuhimedi, Naina Atri, and Lorrie Faith Cranor. Crying wolf: An empirical study of ssl warning effectiveness. In *USENIX Security Symposium*, pages 399–416, 2009.

[84] Brian B. Anderson, Anthony Vance, C. B. Kirwan, J. L. Jenkins, and David Eargle. From warning to wallpaper: Why the brain habituates to security warnings. *Journal of Management Information Systems*, 33(3):713–743, 2016.

[85] Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. Secrets and likes: The drive for privacy and the difficulty of achieving it in the digital age. *Journal of Consumer Psychology*, 30(3):736–758, 2017.

[86] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2013.

[87] H. Song, M. Kim, D. Park, and J. G. Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:1908.11611*, 2019.

[88] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[89] Anne Adams and Martina Angela Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[90] Cormac Herley. So long, and no thanks for the externalities: The rational rejection of security advice by users. In *New Security Paradigms Workshop (NSPW)*, pages 133–144, 2009.

[91] Janyce Wiebe and Theresa Wilson. Annotating subjective expressions in text. *Language Resources and Evaluation*, 2005.

[92] L. Ross and J. McCarthy. Measuring user risk perception in mobile ecosystems. *IEEE Security & Privacy*, 2017.

[93] 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 *USENIX Security Symposium*, pages 531–548, 2018.

[94] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society*, 28(1):20–28, 1979.

[95] Victor Sheng and Foster Provost. Get another label? improving data quality via multiple labels. In *KDD*, 2008.

[96] Curtis G. Northcutt. Pervasive label errors in deep learning datasets. *AAAI Conference on Artificial Intelligence*, 2021.

[97] Bhaskar Pratim Sarma, Ninghui Li, Carrie Gates, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. Android permissions: A perspective combining risks and benefits. In *ACM Symposium on Access Control Models and Technologies*, pages 13–22, 2012.

[98] Y. Zhang and J. Huang. Understanding android permission misuse. *IEEE Transactions on Information Forensics and Security*, 2016.

[99] L. Zhang and M. Zhao. Spectral methods for risk classification. *Pattern Recognition*, 2015.

[100] S. Ray. Sampling bias in mobile security datasets. *ACM Computing Surveys*, 2021.

[101] Eibe Frank and Mark Hall. Simple logistic regression in weka. *Machine Learning*, 2001.

[102] K. Kaseb. Understanding user permissions in android. Medium, February 2022.

[103] Termly. Google play store privacy policy requirements. Termly, 2024.

[104] Cookie Script. New google play store privacy policy requirements. Cookie Script, November 2022.

[105] Android Developers. Google play policies. Android Developers, 2024.

[106] Android Developers. Restrict interactions with other apps. Privacy, February 2025.

[107] Material Design. Android permissions. Material Design, 2024.

[108] Android Developers. Privacy. Android Developers, 2024.

[109] Google Research. The android platform security model. Google Research, 2024.

[110] Google for Developers. Android security and privacy. Google for Developers, 2024.

[111] Android Open Source Project. Android security. Android Open Source Project, December 2024.

[112] Google for Developers. Introduction to the privacy sandbox on android. Google for Developers, March 2025.

[113] Adjust Help Center. Google play data safety. Adjust Help Center, 2024.

[114] Android Open Source Project. Application sandbox. Android Open Source Project, 2024.

[115] Forter. The android manifest. *Forter Blog*, 2024.

[116] Android Developers. Security checklist. Android Developers, June 2025.

[117] Stack Overflow. How to apply restrictions like set a timelimit for another apps, block certain urls from my development app in mobile devices? Stack Overflow, April 2023.

[118] GeeksforGeeks. How to add manifest permission to an android application? GeeksforGeeks, July 2025.

[119] M. Li and K. Thompson. Adaptive vulnerability detection in mobile applications using deep reinforcement learning. *IEEE Access*, 9:145632–145647, 2021.

[120] R. Patel and M. Shah. Vulnerability detection on mobile applications using state machine learning. In *MobiSys 2018*, pages 234–245, 2018.

[121] OWASP Foundation. Owasp mobile application security verification standard (masvs). *Test*, 3(3):3, 2024.

[122] OWASP Foundation. Owasp mobile top 10 security risks. *Test*, 3(3):3, 2024.

[123] National Information Assurance Partnership (NIAP). Niap protection profile for mobile devices. *Test*, 3(3):3, 2024.

[124] PCI Security Standards Council. Pci mobile payment security guidelines. *Test*, 3(3):3, 2024.

[125] U.S. Department of Health and Human Services. Health insurance portability and accountability act (hipaa). *Test*, 3(3):3, 2024.

[126] J. Brannon. User perception and risk awareness in mobile privacy systems, 2025. Unpublished manuscript.

[127] Y. Li, T. Zhang, and J. Chen. A novel risk assessment framework for mobile apps based on the combination of ensemble machine learning and fuzzy logic. *IEEE Access*, 11:123456–123470, 2023.

[128] M. AlGhamdi, S. Alzahrani, and R. Khan. A risk assessment framework for mobile apps in mobile cloud computing environments. *Future Generation Computer Systems*, 146:23–37, 2023.

[129] D. Mathews and P. Novak. Llbezpeky: A data-driven approach to cyber risk threshold optimization. *Journal of Cybersecurity Research*, 2024.

[130] R. Bridges, T. Glass-Vanderlan, M. Iannacone, and M. Vincent. Selecting thresholds for cybersecurity risk models: A data-driven approach. In *IEEE Symposium on Security and Privacy*, 2017.

[131] A. Smith and H. Liu. Credit risk threshold calibration: Lessons for machine learning-based risk models. *Expert Systems with Applications*, 219:119680, 2023.

[132] Curtis G. Northcutt et al. Confident learning: Estimating uncertainty in dataset labels. In ", 2021.

[133] X. Li and Y. Y. Hybrid latent-structural and contextual–semantic learning for risk detection. In *ICML/Risk Analysis Conference*, 2020.

[134] Alessandra Gorla, Ivan Tavecchia, Florian Gross, and Andreas Zeller. Checking app behavior against app descriptions. In *International Conference on Software Engineering (ICSE)*, pages 1025–1035, 2014.

[135] 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.