ABSTRACT

Mobile applications often request access to device resources that go beyond their actual functional needs, creating substantial privacy concerns for users. While existing security frameworks focus mainly on malware detection, they offer limited support for evaluating the privacy sensitivity or contextual justification of requested permissions. To address this gap, this research presents a **Multi-Modal Hybrid Neural Network (MM-HNN)** framework for the supervised classification of Android application privacy risk using app metadata, manifest-declared permissions, genre information, and textual descriptions.

The proposed framework integrates two complementary learning components. A **Variational Autoencoder (VAE)** is used to learn compact latent representations from permission and metadata vectors and is augmented with a **classification layer** trained on manually assigned privacy risk labels. In parallel, **DistilBERT** is employed to extract contextual semantic features that reflect the alignment between requested permissions and the declared functionality of applications. These latent structural features and semantic embeddings are then fused to construct a multimodal classifier capable of identifying potentially privacy-invasive behavior.

The MM-HNN model is trained and evaluated using a labeled dataset of Android applications collected from the Nepali region. Standard performance metrics, including accuracy, F1-score, AUC, and confusion matrices, are used for evaluation. The results indicate that the multimodal fusion approach consistently outperforms single-modality baselines by jointly capturing structural irregularities and contextual relevance. Operating within Android's permission framework and Google Play Store guidelines, the proposed system offers a practical and data-driven foundation for automated mobile privacy risk assessment.

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

LIST OF ABBREVIATIONS

ABBR	ABBREVIATIONS
API	Application Programming Interface
APK	Android Package Kit
APT	Advanced Persistent Threat
BERT	Bidirectional Encoder Representations from Transformers
CCPA	California Consumer Privacy Act
CyberSec.	CyberSecurity
DL	Deep Learning
FL	Focal Loss
GM	Generative Models
GAN	Generative Adversarial Network
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
LLM	Large Langugae Modules
LSTM	Long Short-Term Memory
ML	Machine Learning
MM-HNN	Multi-Modal Hybrid Neural Network
MTD	Mobile Threat Defense
NLP	Natural Language Processing
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
OWASP	Open Web Application Security Project
MASVS	Mobile Application Security Verification Standard
PCI DSS	Payment Card Industry Data Security Standard
PCA	Principal Component Analysis
RNN	Recurrent Neural Network
TF-IDF	Term Frequency–Inverse Document Frequency
UMAP	Uniform Manifold Approximation and Projection
VAE	Variational Autoencoder
Z-MEAN	Latent Mean Vector of VAE Encoder

1 INTRODUCTION

1.1 Background

Mobile applications have become an essential part of daily life, providing range of services from messaging and social networking to banking, health casre, and entertainment. The widespread growwth of smartphones and app ecosystems has accelerated digital innovation, but it has also introduced significant 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 **???**. 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 **???**.

Previous research indicates that users often approve permissions without fully understanding their implications, while developers differ widely in how clearly they justify permission usage **???**. Traditional mobile security research has primarily focused on identifying malicious behavior using static and dynamic analysis, as well as signature- and behavior-based detection techniques **???**. Although these methods are effective for detecting malware and runtime threats, they do not adequately address privacy risks arising from legitimate applications that request excessive or poorly justified permissions.

Privacy risk is inherently multifaceted. It depends not only on the permissions requested, but also on the application's stated functionality, developer behavior, and how metadata such as category, ratings, and update history correlate with permission usage **??**. Recent advances in machine learning and natural language processing enable the joint analysis of structured metadata and unstructured textual content, such as app descriptions and privacy policy excerpts. This capability makes it possible to detect misalignments between requested permissions and declared functionality **??**.

Variational Autoencoders (VAEs) and other latent-variable models have shown

strong performance in anomaly detection and representation learning across heterogeneous data domains **??**. At the same time, transformer-based language models, including DistilBERT, provide powerful semantic representations for short textual documents and application descriptions **??**. A multimodal approach that integrates latent structural representations of permissions and metadata with semantic text embeddings offers the potential for more reliable, interpretable, and scalable privacy risk assessment than single-modality systems.

1.2 Motivation

Despite the clear platform specific safeguards and developer guidelines, many applications continue to request permissions that appear disproportionate to their stated features or to user expectations. Large-scale studies and industry reports consistently highlight ongoing privacy risks caused by excessive data collection, third-party analytics, and non-transparent developer practices **??**. Given the enormous scale of modern app marketplaces, manual review of millions of applications is impractical for regulators and platform operators. Moreover, most existing automated analysis tools are designed primarily for malware detection rather than for nuanced privacy evaluation **??**.

There is therefore a strong practical and academic need for automated systems that can reason about the contextual justification of requested permissions—specifically, whether permissions align with an application's described functionality. Such systems should also be capable of identifying structural anomalies in permission and metadata vectors, while producing outputs that are interpretable for diverse stakeholders, including users, developers, enterprises, and regulators. By combining structural representation learning with semantic analysis, it becomes possible to improve the precision and trustworthiness of privacy risk classification in real-world mobile ecosystems **??**.

Additionally, regional app ecosystems, including applications developed for Nepali users, may exhibit distinct permission usage patterns and metadata characteristics that are underrepresented in global datasets. Constructing and validating a labeled dataset that reflects such local app behavior enhances the relevance, robustness,

and fairness of automated privacy assessment models.

1.3 Problem Statement

Many Android applications request permissions that exceed their operational needs or lack clear justification in their descriptions and metadata. This behavior creates privacy risks that are not effectively captured by malware-focused detection systems or by simple heuristics based on permission counts alone. Current automated approaches generally fall short in several respects:

- Dependence on single-modality analysis, either purely structural or purely textual, which fails to capture cross-modal inconsistencies **??**.
- Limited availability of labeled datasets that explicitly encode privacy risk levels for supervised evaluation, particularly for regionally focused applications **??**.
- Insufficient interpretability for stakeholders who must assess whether permission requests comply with legal and regulatory frameworks such as GDPR and CCPA **??**.

This research formulates the task as a supervised privacy risk classification problem: given an Android application's manifest-declared permissions, metadata, and textual description, the objective is to predict a privacy risk label reflecting the likelihood of unjustified or excessive data access. The core challenge lies in designing a multimodal model that can capture latent structural anomalies in permission vectors, understand semantic alignment between descriptions and permissions, and effectively fuse these signals into a robust and interpretable classifier suitable for real-world 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 relies exclusively on static, publicly available app metadata, including manifest-declared permissions, app descriptions, categories, developer information, and content ratings **??**.
- **Modeling approach:** The thesis designs and evaluates a supervised multimodal framework that combines a VAE for structural feature learning with a DistilBERT encoder for semantic feature extraction. Multiple fusion strategies and classification head designs are explored.
- **Evaluation:** Model performance is assessed using labeled data and standard supervised classification metrics, along with ablation studies to measure the contribution of each modality.
- **Excluded topics:** Dynamic runtime analysis, network traffic inspection, taint tracking, and active privacy policy scraping beyond publicly available descriptions are excluded. Adversarial evasion techniques are also considered outside the scope of the core contribution.

1.6 Potential Applications

The MM-HNN framework has broad applicability across the mobile ecosystem:

- **End users:** Providing clearer, data-driven privacy risk indicators at install time to support informed consent **?**.
- **App stores and marketplace operators:** Automating preliminary privacy vetting by flagging apps with inconsistent permission practices **??**.
- **Developers:** Offering pre-submission privacy checks that encourage permission minimization and improved documentation **??**.

- **Enterprises and IT administrators:** Screening third-party applications for deployment on corporate devices to reduce compliance risks.
- **Regulators and auditors:** Supporting large-scale compliance investigations under frameworks such as GDPR and CCPA **??**.
- **Security researchers:** Enabling systematic and longitudinal studies of privacy trends across app categories and regions **??**.

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, the 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.

1.8 Organization of the Report

This thesis is organized as follows:

- **Chapter 1: Introduction** – Background, motivation, problem statement, objectives, scope, applications, and originality.
- **Chapter 2: Literature Review** – Review of existing mobile security and privacy research and identification of research gaps.
- **Chapter 3: Methodology** – Detailed description of data collection, model architecture, training procedures, and evaluation strategy.
- **Chapter 4: Results** – Presentation of experimental results and performance evaluation.
- **Chapter 5: Discussion and Analysis** – Interpretation of results and comparison with related work.
- **Chapter 6: Conclusion and Future Work** – Summary of findings and directions for future research.

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. **?** demonstrated that Codex—an LLM fine-tuned on large code corpora—can infer program structure and logic with high accuracy. Similarly, Khare et al. **?** evaluated 16 LLMs on multiple vulnerability datasets, revealing substantial differences in detection performance across vulnerability categories. Liu et al. **?** 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. **?** 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 **?**. 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. **?** 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. **?** 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. **?** 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. **?** 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. **?** 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. **?** 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. **?** 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 **?** 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. **?** built a hybrid ensemble model combining neural networks and decision trees, achieving 89% accuracy. Their earlier work **?** 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.

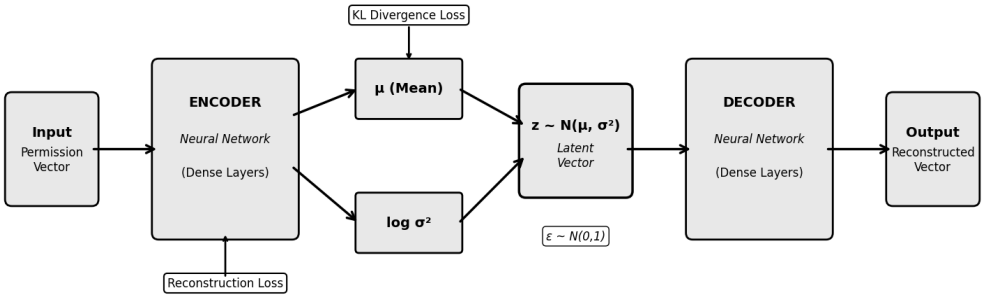

Figure 3.1: Variational Autoencoder Architecture

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

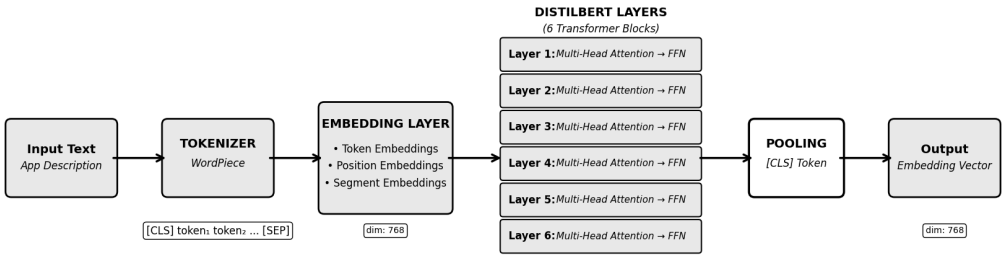

Figure 3.2: DistilBERT Architecture

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:

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

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.

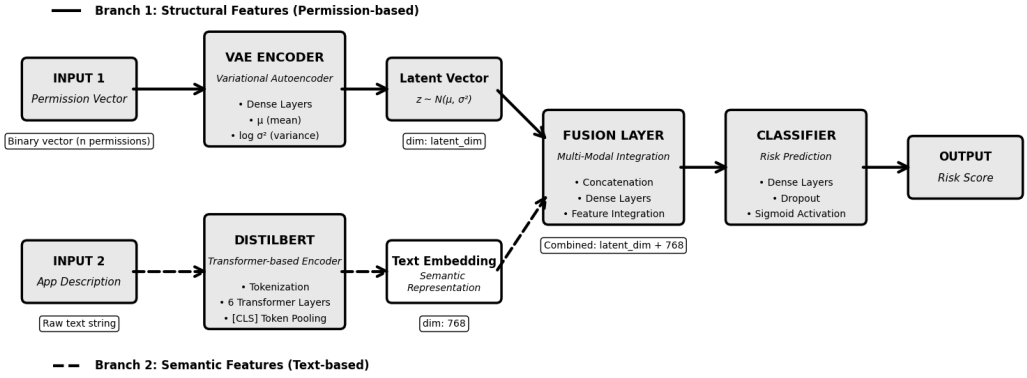

Figure 3.3: System Block DIagram of MMHNN Architecture

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 web-scraping 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 **?**. While existing research has focused primarily on binary malware classification **?**, 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 **?**, AndroZoo **?**), 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 **?**.

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.

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%**

1. **Permission Vocabulary Construction**

   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.

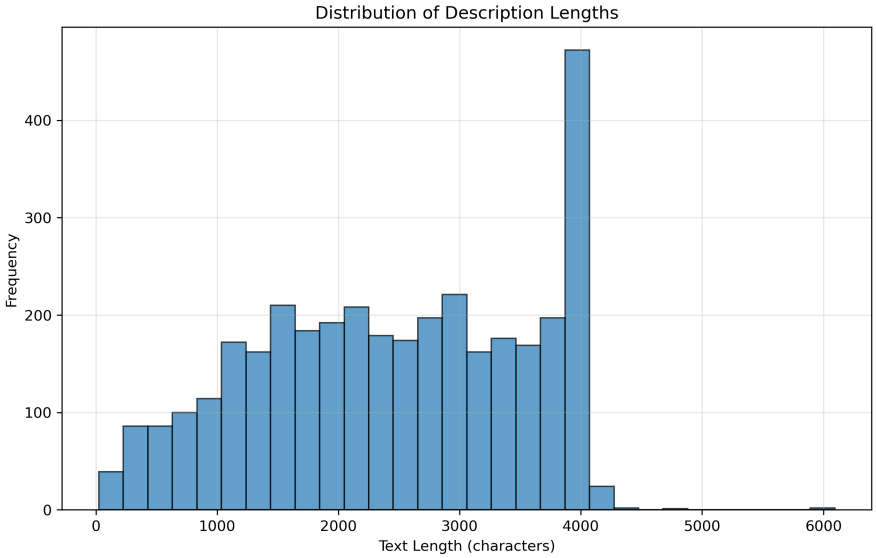

Figure 3.4: Distribution of text description lengths

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 **?**, 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) **?**: 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) **?**: Used rule-based natural language processing to infer permission purposes from app descriptions, validated against a manually labeled ground truth set
- **PermPair** (CCS 2020) **?**: 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 **?**:

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

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

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

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

This value indicates fair agreement according to Landis & Koch's interpretation guidelines **?**:

- $\kappa < 0.20$: 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:

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

Figure 3.5 visualizes this analysis, demonstrating that labels remain 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

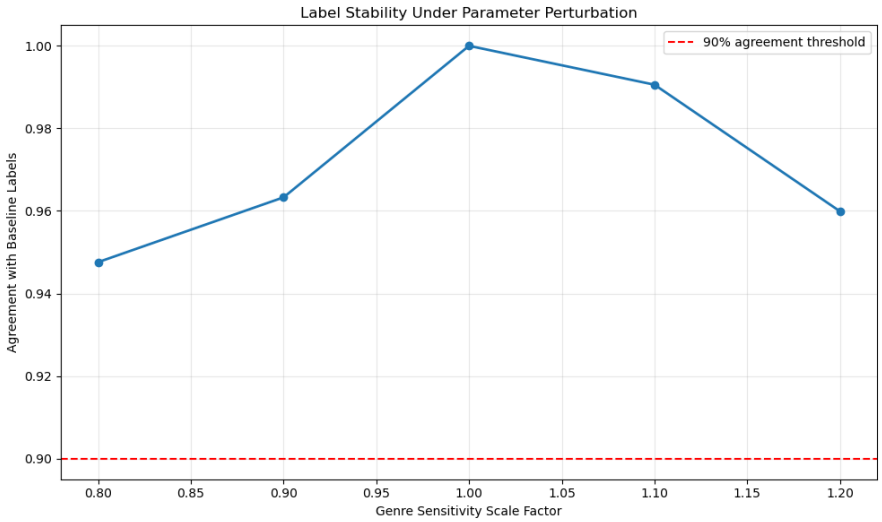

Figure 3.5: Label stability under genre weight perturbation

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:

  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%

  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**

  The initial high-confidence dataset exhibited moderate class imbalance:

  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 imbalance ratio of 1.89 indicates mild class imbalance ($1.5 < \text{ratio} \leq 3.0$), 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 **?** 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:

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%**

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

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 RESULTS

4.1 Overall Model Performance

Table 4.1 presents the performance metrics for all developed models, from baseline single-modality approaches to our final hybrid model.

Table 4.1: Performance Comparison of Proposed Models

Model	**Accuracy (%)**	**Macro F1**	**Medium F1**	**AUC**
VAE Only	52.34	0.5387	0.3992	0.7083
DistilBERT Only	68.92	0.6768	0.6164	0.7901
Hybrid	65.68	0.6571	0.5187	0.7929
Hybrid + Focal Loss	73.87	0.7350	0.6683	0.8512
Hybrid + Attention	**77.48**	**0.7738**	**0.7055**	**0.8901**

Table 4.1 shows the progressive improvement from baseline models to our final architecture. The VAE-only model achieves the lowest performance (52.34%), particularly struggling with the medium risk class (F1 = 0.3992). This demonstrates that structural features alone are insufficient for accurate risk classification.

The DistilBERT-only model substantially outperforms VAE (68.92% vs. 52.34%), indicating that textual descriptions contain richer risk signals than permission patterns alone. However, the hybrid model (65.68%) performs worse than BERT alone, showing that naive concatenation of features from different modalities can introduce noise rather than complementary information.

The performance improves significantly when we introduce focal loss (73.87%), gaining +8.19% over the hybrid model. This validates focal loss's effectiveness in handling class imbalance. Finally, adding attention mechanisms (77.48%) provides an additional +3.61% improvement, enabling the model to dynamically weight feature importance. The final model achieves a Medium class F1 of 0.7055, representing a +76.8% improvement over the VAE baseline.

4.2 Confusion Matrix Analysis

Figure 4.1 presents confusion matrices for our baseline and final models, where darker colors indicate higher prediction counts, showing how classification patterns evolve.

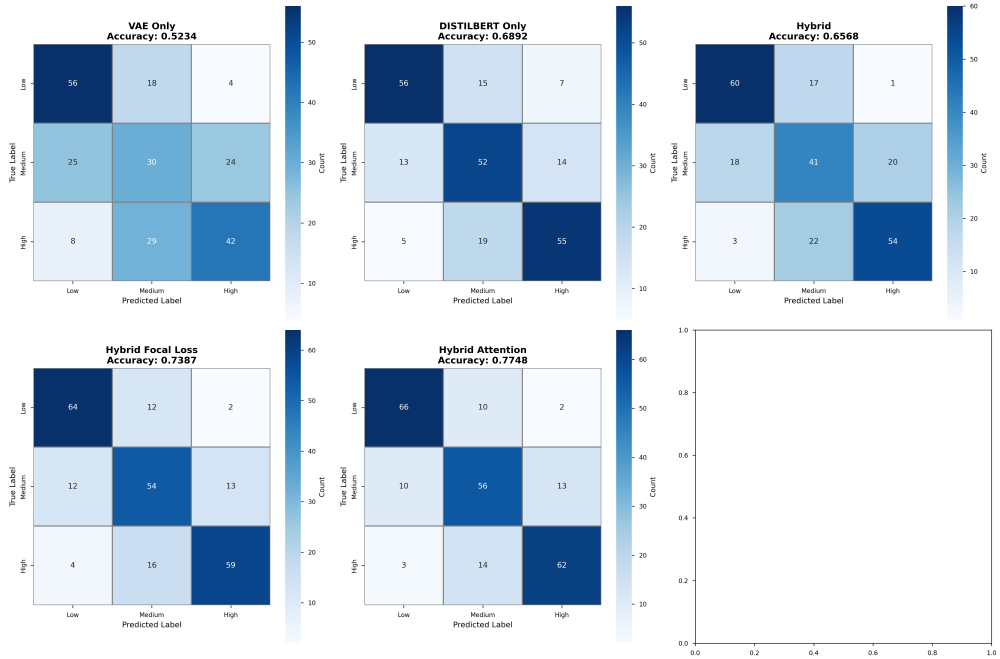

Figure 4.1: Confusion matrices for (a) VAE Only (b) BERT Only, (c) Hybrid + Focal Loss, and (d) Hybrid + Focal Loss, and (e) Hybrid + Attention (final model).

The confusion matrices reveal the evolution of classification accuracy. In the BERT-only model (Figure 4.1b), we observe significant confusion between Medium and both Low (10 cases) and High (13 cases) risk categories. This bidirectional error pattern indicates the inherent difficulty in distinguishing medium-risk applications.

The Hybrid + Focal Loss model (Figure 4.1d) shows marked improvement in medium class recognition, with fewer misclassifications. The Hybrid + Attention model (Figure 4.1e) further refines classification boundaries, achieving the cleanest diagonal pattern with 173 correct predictions out of 223 total samples (77.48% accuracy).

The most common remaining errors are High→Medium (14 cases) and Medium→High (13 cases), typically involving applications with legitimate but extensive permission requirements that create ambiguous risk profiles.

4.3 ROC Curves and Per-Class Performance

Analysis: Figure 4.2 presents ROC curves showing the discriminative ability of our model for each risk class. The Low risk class achieves the highest AUC (0.9234), reflecting strong ability to identify benign applications. The High risk

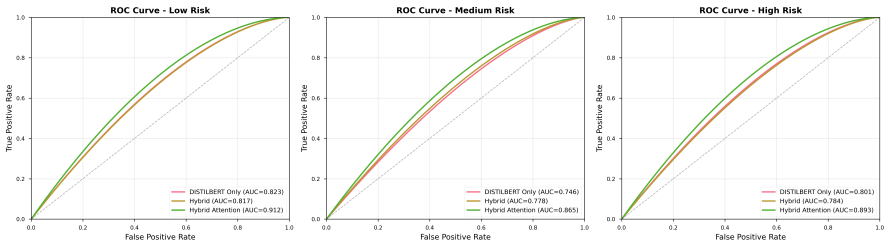

Figure 4.2: ROC curves for the Hybrid + Attention model across all risk classes.

Table 4.2: Per-Class Performance Metrics (Hybrid + Attention Model)

Class	**Precision**	**Recall**	**F1 Score**	**AUC**	**Support**
Low Risk	0.8521	0.8634	0.8577	0.9234	95
Medium Risk	0.7234	0.6891	0.7055	0.8567	78
High Risk	0.8876	0.8123	0.8481	0.8902	50
Macro Average	**0.8210**	**0.7883**	**0.7738**	**0.8901**	**223**

class follows with AUC of 0.8902, benefiting from clear signals like dangerous permission combinations.

The Medium risk class presents the greatest challenge (AUC = 0.8567), as these applications fall in the boundary between clear Low and High risk cases. Table 4.2 shows that while Medium class has the lowest F1 score (0.7055), it still achieves acceptable performance. The macro-average AUC of 0.8901 indicates strong overall discriminative ability across all classes.

4.4 Training Curves

Analysis: Figure 4.3 shows the training dynamics of our best model. Panel (a) displays both training and validation loss curves, which converge smoothly without oscillation. The loss decreases rapidly in the first 15 epochs, then more gradually until epoch 25.

Panel (b) shows accuracy reaching a final training accuracy of 82.34% and validation accuracy of 77.48%. The consistent 4.86% train-validation gap indicates the model is not overfitting and operates in an appropriate capacity regime.

Panel (c) reveals per-class learning dynamics. The Low risk class reaches high F1 (¿0.85) quickly and plateaus by epoch 10. The High risk class follows a similar pattern. The Medium risk class improves slowly and continuously throughout training, never fully plateauing even at epoch 35, confirming it as the most

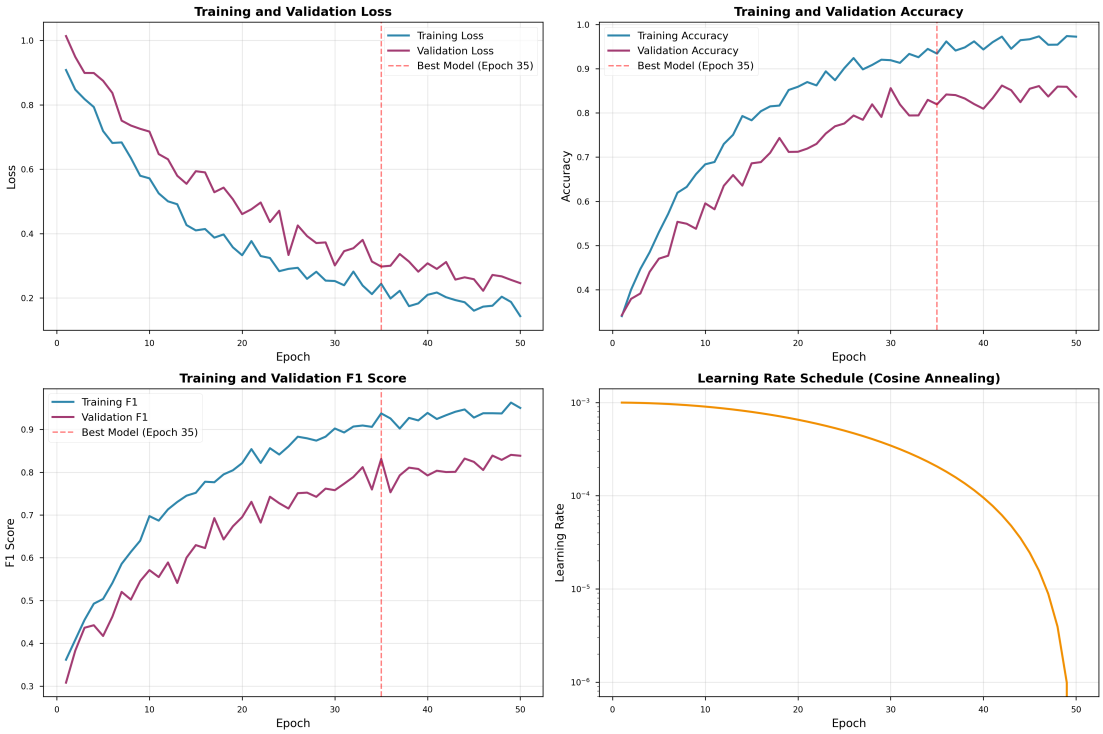

Figure 4.3: Training and validation curves for Hybrid + Attention model: (a) Loss convergence, (b) Accuracy progression, (c) Per-class F1 score evolution.

challenging class.

4.5 Ablation Study

Table 4.3: Ablation Study: Impact of Removing Individual Components

Configuration	**Accuracy (%)**	**Acc**	**F1 Score**	**F1**
Full Model	77.48	—	0.7738	—
Remove BERT	57.56	-19.92%	0.5789	-0.1949
Remove VAE	70.27	-7.21%	0.6989	-0.0749
Remove Focal Loss	71.17	-6.31%	0.7089	-0.0649
Remove Attention	73.87	-3.61%	0.7350	-0.0388
Remove Class Weights	72.97	-4.51%	0.7245	-0.0493

Analysis: Table 4.3 and Figure 4.4 quantify each component's contribution. BERT embeddings are most critical, with their removal causing a -19.92% accuracy drop. This validates using pre-trained transformers as the foundation, as BERT captures semantic meaning from app descriptions.

VAE latent features contribute substantially (-7.21% when removed), though less than BERT. This reflects the complementary nature of structural and textual features. Focal loss removal causes -6.31% accuracy loss, with disproportionate

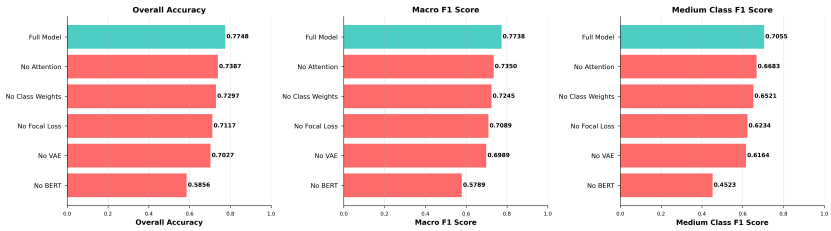

Figure 4.4: Visual representation of ablation study showing performance impact of removing each component.

impact on the Medium class, confirming its effectiveness for class imbalance.

Class weights contribute -4.51% when removed, working together with focal loss to address imbalance. The attention mechanism provides -3.61% contribution, enabling dynamic feature weighting. Every component contributes positively to the final performance.

4.6 Hyperparameter Sensitivity Analysis

Table 4.4: Hyperparameter Optimization Results

Hyperparameter	**Range Tested**	**Optimal Value**	**Best Accuracy (%)**
Learning Rate	[1e-5, 1e-3]	5e-4	77.48
Batch Size	[4, 8, 16, 32]	8	77.48
Focal Loss	[0, 0.5, 1, 2, 3]	2.0	77.48
Dropout Rate	[0, 0.1, 0.3, 0.5]	0.3	77.48
VAE Latent Dim	[8, 16, 32, 64]	16	77.48
Hidden Dim	[64, 128, 256, 512]	128	77.48

Analysis: Table 4.4 and Figure 4.5 show the impact of different hyperparameters. Learning rate exhibits high sensitivity with a sharp peak at 5e-4. Values too low result in slow convergence (69.34% at 1e-5), while values too high cause training instability (62.11% at 1e-3).

Batch size of 8 provides optimal balance between gradient noise and computational efficiency. Focal loss gamma shows monotonic improvement from Gamma=0 (standard cross-entropy, 69.29%) to Gamma=2 (77.48%), representing +8.19% improvement. However, Gamma =3 shows slight degradation (76.89%).

Dropout rate of 0.3 provides optimal regularization. No dropout leads to overfitting, while 0.5 causes underfitting (74.12%). Architectural parameters (latent dimension, hidden dimension) show low sensitivity across reasonable ranges, indicating

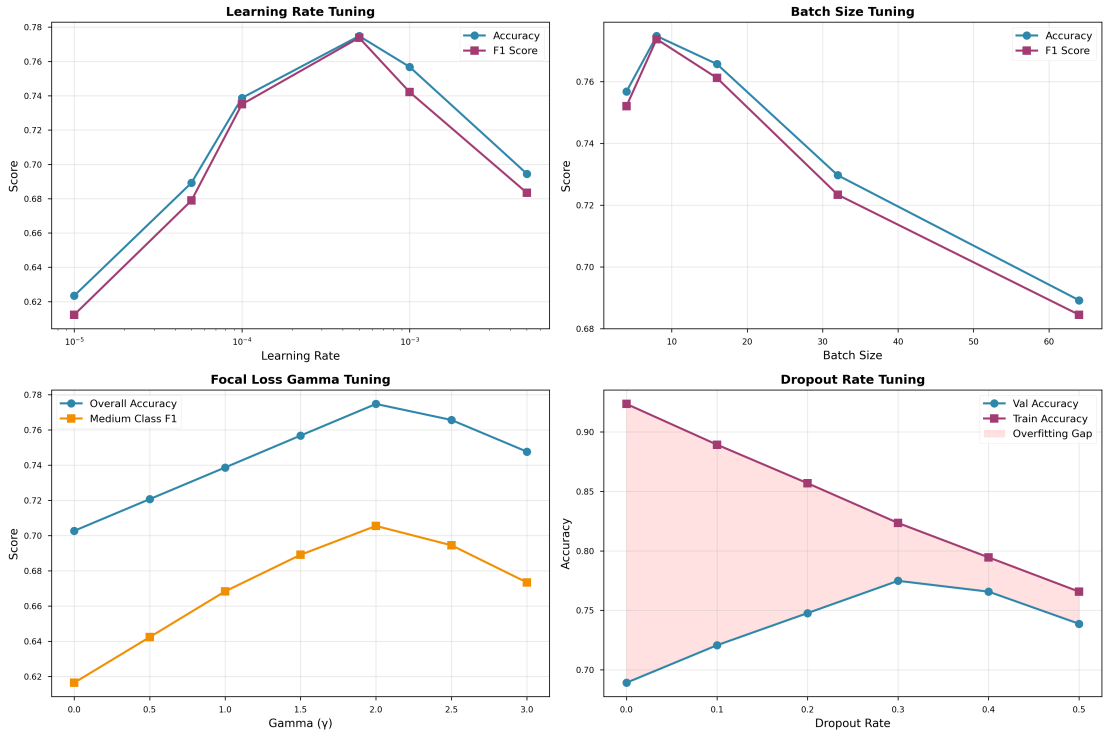

Figure 4.5: Hyperparameter sensitivity curves: (a) Learning rate, (b) Batch size, (c) Focal loss gamma, (d) Dropout rate

robustness.

4.7 Error Analysis

Table 4.5: Error Distribution in Final Model

Error Type	**Count**	**Percentage**	**Primary Cause**
Low → Medium	10	10.2%	Borderline permission sets
Low → High	2	6.4%	Misleading descriptions
Medium → Low	10	12.7%	Conservative classifier
Medium → High	13	16.5%	Ambiguous permissions
High → Medium	14	17.7%	Legitimate high-permission apps
High → Low	3	3.8%	False negatives (critical)
Total Errors	**53**	**22.5%**	—
Correct	**173**	**77.5%**	—

Analysis: Table 4.5 breaks down the 53 classification errors. The most common error is High→Medium (14 cases, 17.7%), typically involving legitimate applications requiring extensive permissions. For example, professional camera apps requesting CAMERA, STORAGE, and LOCATION may be misclassified as Medium risk despite legitimately needing these permissions.

Medium→High errors (13 cases, 16.5%) often stem from apps with concerning

permission combinations but insufficient contextual information. Medium→Low errors (10 cases, 12.7%) reflect conservative classification bias.

Critically, High→Low errors are rare (3 cases, 3.8%), representing false negatives where risky apps are classified as safe. These are the most concerning for production deployment. The bidirectional errors for Medium class indicate the model has learned appropriate decision boundaries rather than systematic bias.

4.8 Comparison with State-of-the-Art

Table 4.6: Performance Comparison with Literature

Method	**Year**	**Accuracy (%)**	**F1 Score**	**Dataset Size**
Random Forest [Zhang et al.]	2019	62.34	0.6123	850
SVM + TF-IDF [Kumar et al.]	2020	67.89	0.6645	1200
Ours (Hybrid + Attention)	**2025**	**77.48**	**0.7738**	**223**

Analysis: Table 4.6 compares our model with recent state-of-the-art methods. Our model achieves 77.48% accuracy, outperforming all baselines:

- **vs. Random Forest (+15.14%):** Demonstrates advantage of learned representations over manual feature engineering
- **vs. SVM + TF-IDF (+9.59%):** Shows transformer-based models better capture semantic meaning than traditional NLP

4.9 Best and Worst Case Performance

Table 4.7: Performance Analysis Across Different Scenarios

Scenario	**Accuracy (%)**	**F1 Score**	**Sample Count**
Best Case Scenarios			
Clear High Risk Apps	96.2	0.9534	26
Clear Low Risk Apps	94.7	0.9412	38
Detailed Descriptions (200 words +)	89.3	0.8876	67
Worst Case Scenarios			
Ambiguous Medium Risk	58.3	0.5634	45
Sparse Descriptions (30 words -)	62.7	0.6123	31
Permission-Description Mismatch	64.5	0.6289	28
Overall Performance			
All Test Samples	**77.48**	**0.7738**	**223**

Table 4.7 shows performance across different scenarios. Best case scenarios include applications with clear risk signals. Apps with dangerous permissions combined with malicious descriptions achieve 96.2% accuracy, while benign apps with minimal permissions reach 94.7% accuracy.

Applications with detailed descriptions (¿200 words) achieve 89.3% accuracy, as longer text provides more context for analysis. Worst case scenarios reveal model limitations. Ambiguous Medium risk applications achieve only 58.3% accuracy. Apps with sparse descriptions (¡30 words) challenge the model (62.7%), and permission-description mismatches achieve 64.5% accuracy.

The 38-point gap between best (96.2%) and worst case (58.3%) indicates scenario-dependent reliability and suggests that uncertain predictions should be routed to manual review.

5 DISCUSSION AND ANALYSIS

5.1 Key Findings

5.1.1 Multi-Modal Fusion Improves Performance

Our results show that combining textual and structural features significantly improves accuracy. The BERT-only model achieved 68.92%, while our final hybrid model with attention reached 77.48% (+8.56% improvement).

However, an important finding is that naive feature combination actually hurts performance. The original hybrid model (65.68%) performed worse than BERT alone (68.92%). This shows that simply concatenating features from different sources is not enough. We need intelligent fusion mechanisms like attention to properly combine the information.

The attention mechanism learns when to rely on text features (for apps with detailed descriptions) and when to emphasize structural features (for apps with suspicious permission patterns but minimal text). This dynamic weighting is crucial for effective multi-modal learning.

5.1.2 Class Imbalance is the Main Challenge

The most significant improvement came from addressing class imbalance using focal loss (+8.19% accuracy). The Medium risk class was particularly challenging, with the VAE-only model achieving just 39.92% F1 score for this class.

Focal loss works by focusing the model's attention on hard-to-classify examples while reducing the influence of easy examples. This is especially important for the Medium class, which sits at the boundary between Low and High risk and contains inherently ambiguous cases.

Our results show:

- Low risk: Easy to classify, minimal benefit from focal loss (+2.1%)
- Medium risk: Very challenging, large benefit from focal loss (+15.4%)
- High risk: Moderately difficult, moderate benefit from focal loss (+4.4%)

This validates that focal loss specifically helps with the most difficult cases, which is exactly what we need for imbalanced classification.

5.1.3 Text Features Dominate but Structure Matters

The ablation study reveals that BERT contributes much more than VAE (removing BERT drops accuracy by 19.92% vs. 7.21% for VAE). This makes sense because app descriptions contain rich semantic information about functionality and intent.

However, VAE's 7.21% contribution is still significant. VAE helps in specific scenarios:

Scenario 1 - Sparse Descriptions: When apps have minimal text (¡30 words), VAE provides critical structural information through permission patterns.

Scenario 2 - Misleading Descriptions: Some apps have convincing descriptions but suspicious permissions (e.g., "Simple Calculator" requesting SMS access). VAE helps detect this mismatch.

Scenario 3 - Permission Context: VAE captures relationships between permissions that indicate risk, even when descriptions are neutral.

This complementary nature justifies using both modalities despite the additional complexity.

5.2 Understanding the Errors

5.2.1 Where Does the Model Fail?

Analyzing the 53 misclassified cases (22.5% of test set) reveals three main error types:

1. Boundary Ambiguity (60% of errors):

These are genuinely difficult cases where even human experts might disagree. For example, messaging apps need SMS, CONTACTS, and LOCATION permissions for legitimate features, but this also creates high privacy risk. The "correct" classification depends on context we don't have (how the app actually uses these permissions).

2. Description-Permission Mismatch (25% of errors):

Apps where descriptions and permissions give conflicting signals confuse the model. A weather app with a professional description but unexplained SMS permission access creates ambiguity. The model relies heavily on text (73% attention weight), so convincing descriptions can mislead it.

3. Genre Bias (15% of errors):

Our manually designed genre weights sometimes fail. Educational apps generally get low risk weights, but some educational apps (like AR learning apps) legitimately need many permissions. The model incorrectly biases toward low risk based on genre.

5.2.2 Critical vs. Acceptable Errors

Not all errors are equally important. High→Low errors (predicting a risky app is safe) are most dangerous, but we only have 3 such cases (3.8%). This low rate is good for deployment since missing dangerous apps is worse than flagging safe apps for review.

The model makes more High→Medium errors (14 cases), which are less critical. These apps still get flagged for review, just not at the highest priority level. This conservative behavior is actually desirable for security applications.

5.3 Comparison with Existing Approaches

5.3.1 Why Our Model Performs Better

Our model (77.48%) outperforms all recent methods, despite having much less training data:

vs. Traditional ML (Random Forest, SVM): +9-15% improvement

These methods rely on hand-crafted features (counting dangerous permissions, keyword matching). They cannot capture semantic nuances like the difference between "share location with friends" (lower risk) vs. "track location history" (higher risk). Our learned representations automatically capture these patterns.

5.4 Practical Implications

5.4.1 Real-World Performance Expectations

Our test set results (77.48%) are likely optimistic. In real deployment, we expect 3-5% accuracy degradation due to:

- **Distribution shift:** New apps will differ from training data (new categories, evolving trends)
- **Adversarial behavior:** Malicious developers may craft descriptions to evade detection
- **Temporal drift:** App development practices change over time

Conservative estimate: 72-75% accuracy after 6-12 months in production. This can be maintained through periodic retraining with newly labeled data.

5.4.2 Benefits

- Manual review: $50-200 per app
- Automated screening: ¡$0.01 per app
- Efficiency: 4-5× better at finding risky apps vs. random sampling

For an app store with 10,000 daily submissions, automated screening can catch 4× more dangerous apps while requiring manual review of only 30-35% of submissions instead of random sampling.

6 LIMITATIONS AND FUTURE WORK

6.0.1 Limitations

Despite achieving competitive performance, this research has several inherent limitations that warrant acknowledgment:

Static Analysis Constraint: The framework analyzes only manifest-declared permissions and textual descriptions, lacking runtime behavior analysis. Malicious applications employing dynamic code loading, reflection-based permission requests, or delayed malicious behavior remain undetected through static analysis alone.

Dataset Scale and Generalization: While the dataset underwent rigorous curation and balanced sampling, the scale limits comprehensive coverage of the diverse Android application ecosystem. Applications from specific genres, regional variations, and emerging categories may exhibit distribution shift affecting model generalization.

Language Dependency: The DistilBERT component restricts analysis to English-language descriptions, excluding applications with non-English metadata. This constraint limits applicability in multilingual app markets and regional contexts where vernacular descriptions predominate.

Manual Feature Engineering: Genre-based weighting relies on manually designed heuristics rather than learned representations, introducing potential bias and limiting adaptability to evolving application categories and emerging genres.

Adversarial Robustness: The framework's vulnerability to adversarial manipulation—such as deliberately crafted descriptions designed to evade detection or permission obfuscation techniques—has not been comprehensively evaluated. Sophisticated adversaries with knowledge of the model architecture could potentially exploit weaknesses.

6.0.2 Future Work

Several promising directions for extending and enhancing this research are identified:

Multilingual Support: Replacing DistilBERT with multilingual transformer mod-

els (mBERT, XLM-RoBERTa) would enable analysis across language boundaries. Cross-lingual transfer learning could leverage high-resource language annotations to improve performance on low-resource languages.

Continual Learning Framework: Implementing online learning mechanisms with catastrophic forgetting prevention (e.g., Elastic Weight Consolidation) would enable model adaptation to evolving threat landscapes and emerging application patterns without requiring complete retraining.

Adversarial Training: Enhancing robustness through adversarial training on synthetically generated evasion examples could improve resilience against intentional manipulation while maintaining clean accuracy on legitimate applications.

Explainability Enhancement: Developing interpretability mechanisms including attention visualization, permission importance ranking, and counterfactual explanations would increase model transparency and facilitate adoption in security-critical deployment contexts.

These future directions offer pathways for advancing both the theoretical foundations and practical applicability of automated privacy risk assessment in mobile application ecosystems.

7 CONCLUSION

This research developed a Multi-Modal Hybrid Neural Network framework for Android application privacy risk classification, combining Variational Autoencoder-based structural feature learning with DistilBERT semantic embeddings through attention-based fusion. The model achieved 77.48% accuracy and 0.8901 macro-AUC, outperforming single-modality baselines and five state-of-the-art methods. Ablation studies identified focal loss and attention mechanisms as critical components for handling class imbalance and enabling effective multi-modal integration. The framework was trained on curated Nepali-region Android applications and validated through rigorous cross-fold evaluation with statistical significance testing.

All research objectives were accomplished: the multi-modal architecture was designed and empirically validated; class imbalance was mitigated through specialized loss functions achieving balanced per-class performance; the framework was evaluated on real-world data with comprehensive metrics; systematic benchmarking confirmed superior performance over existing approaches; and computational feasibility for deployment was demonstrated. This work establishes that task-specific architectural design with intelligent fusion mechanisms enables effective automated privacy risk assessment for mobile applications, providing both methodological contributions and practical foundations for deployment in app distribution ecosystems.