

本科毕业设计(论文)

GRADUATION DESIGN(THESIS)

|  |  |
| --- | --- |
| 题 目： | 移动应用中深度学习模型完整性度量方法的研究 |
| 学生姓名： | 徐贯虹 |
| 指导教师： | 段桂华 |
| 学 院： | 计算机学院 |
| 专业班级： | 信安1701班 |

本科生院制

2021年06月

移动应用中深度学习模型完整性度量方法的研究

摘要

随着智能手机的普及，信息化程度不断深入，作为智能手机主体的移动应用已经渗透到我们生活的方方面面。智能设备可移动、易携带以及方便快捷的特性，推动移动应用程序发展，实现更加强大丰富的功能，其中深度学习受到了青睐，被越来越广泛地使用。但是深度学习与移动应用为我们带来的便利的同时，也带来了安全问题。模型所有者的核心知识产权即深度学习模型，存储于移动设备上，容易被盗窃；用于安全或隐私相关模块的模型，例如线上购物试穿衣服功能、银行卡身份证等线上识别认证，存在被完整性攻击的安全风险。因此，针对于移动应用上深度学习模型的完整性研究有着重大意义。

针对上述问题，本文对移动应用上深度学习模型进行完整性度量。在分析当前移动应用中深度学习模型的安全保护的一些研究成果基础上，提出了对移动应用中深度学习模型的完整性度量方法，能够通过检查动态链接库中特征字符串，识别使用了深度学习的移动应用，动态与静态方法结合，分别基于特征与Frida插桩工具提取移动应用中的深度学习模型，并对模型参数、模型格式、模型使用可能出现完整性安全问题进行分析。实验数据分析结果表明，在移动应用中使用深度学习的安全性不足，移动平台模型保护有待加强。

**关键词：**移动应用 深度学习模型 完整性 逆向工程

Research on Integrity Measurement Methods of Deep Learning Model in Mobile Apps

**ABSTRACT**

With the popularization of smart phones and the deepening of informatization, mobile applications as the main body of smart phones have penetrated into all aspects of our lives. The mobile, easy-to-carry, and convenient features of smart devices promote the development of mobile applications to realize more powerful and rich functions. Among them, deep learning is favored and used more and more widely. But while deep learning and mobile applications bring us convenience, they also bring security issues. The core intellectual property of the model owner is the deep learning model, which is stored on the mobile device and is easy to be stolen; the model used for security or privacy-related modules, such as online shopping to try on clothes, bank card ID and other online identification and authentication may have security risk of integrity attacks. Therefore, research on the integrity of deep learning models on mobile applications is of great significance.

The purpose of this research is to measure the integrity of deep learning models on mobile applications in response to the above problems. Based on the analysis of some research results on the security protection of deep learning models in current mobile applications, this thesis proposes a method for measuring the integrity of deep learning models in mobile applications. It can identify the use of deep learning in mobile applications by checking the characteristic strings in the dynamic link library, extract the deep learning model in the mobile application in both dynamic and static methods respectively, and analyze the integrity and safety issues that may occur in the model parameters, model format, and model use. This research shows that the security of using deep learning in mobile applications is insufficient, and the protection of mobile platform models needs to be strengthened.

**Key words：**Mobile application Deep learning model Integrity Reverse engineering

目录

第1章 引言 1

第2章 研究现状 3

2.1 移动应用中深度学习模型识别 3

2.2 移动应用中机器学习模型提取 4

2.3 针对模型的完整性攻击 5

2.3.1 逃逸攻击 5

2.3.2 投毒攻击 6

2.4 Android逆向 7

2.4.1 Android运行机制 7

2.4.2 Android使用深度学习 8

2.5 环境及工具使用 9

2.6 本章小结 10

第3章 研究方法设计概述 12

3.1 流程概述 12

3.2 识别深度学习应用 12

3.3 提取深度学习模型 13

3.4 完整性度量方法 15

3.5 本章小结 16

第4章 方法实现与结果分析 17

4.1 实验环境与数据集 17

4.2 方法实现 18

4.2.1 识别深度学习应用 18

4.2.2 提取深度学习模型 19

4.2.3 完整性度量 21

4.3 结果分析 22

4.3.1 深度学习应用程序 22

4.3.2 深度学习框架 23

4.3.3 深度学习模型 24

4.3.4 完整性度量分析 25

4.4 本章小结 26

第5章 总结和展望 27

5.1 工作总结 27

5.2 研究展望 27

参考文献 29

致谢 31

# 第1章 引言

移动手机越来越多地渗入到生活的方方面面，据统计移动设备已经占了网络流量的大半，Twitter的使用时间、Amazon的销售更多的是在移动设备上完成的，移动手机无处不在，是最有前途的深度学习（DL）应用平台之一，这也是近年来移动智能的主要动力[1-4]。深度学习蓬勃发展，在移动应用中越来越流行，因其在计算机视觉、自然语言处理以及语音识别中有着优秀的表现，被广泛使用在图像处理、语音识别、文字预测、图片扫描、购物推荐等移动应用中。

2017年，大多数主要供应商都推出了用于智能手机的深度学习框架，或简称为移动深度学习框架。这些框架包括Google（2017年11月）的TensorFlow Lite（TFLite），Facebook的Caffe2（2017年4月），Apple的Core ML（2017年6月），腾讯的ncnn（2017年7月）和百度的MDL（2017年9月）。这些框架具有相同的目标：设备上深度学习推理。相较于使用云端支持智能手机的深度学习，设备上的深度学习推理可以更好地保护用户隐私，消除网络延迟的影响，在互联网连接不佳的情况下仍可继续运行，并且减轻了应用作者支付在云中运行深度学习的费用[1,5]。设备上深度学习在每种类型的移动应用中都越来越流行。例如，在Amazon购物中使用三维人体识别为会员提供试穿衣服的功能，在Alipay支付宝付款应用中使用图片扫描识别身份证，以及手机助手的语音识别等。

但是大多数移动应用的开发者在开发应用时没有考虑到模型存在的安全风险，机器学习模型在设计时未考虑攻击者的存在，尽管移动应用中的深度学习模型在正常工作时能有优秀表现，但现实中可能存在大量恶意用户甚至是攻击者，面临各种不同程度的风险，导致模型无法提供正常服务，或泄露相关隐私信息[6]。深度学习模型运行在数十亿不受信任的移动设备上，未经保护或仅仅简单保护的模型给攻击者留下了极大的可乘之机，攻击者可能对模型的训练参数等进行恶意篡改，从而破坏模型的完整性。

模型泄露的后果非常严重，可能导致针对模型完整性的白盒攻击。模型泄露会造成其所有者的研发投资等成本的损失，不道德竞争者抢占产品优势，更严重地会帮助恶意行为者找到攻击者的输入，混淆通过模型系统，可能导致其产品出现严重故障（例如指纹识别绕过）。

完整性面临的对抗攻击逐渐兴起，引起安全研究人员的关注。现下越来越多人工智能赋能安全的使用，深度学习被用在钓鱼邮件检测、入侵检测、C&C检测、反爬虫、账户泄露检测等安全应用中，其模型完整性一旦被破坏，模型预测结果偏离预期，可能造成的安全问题是更加严重的。

目前为了保护设备上的模型，一些开发人员对其进行加密或混淆处理，或者将它们编译为应用程序代码，然后将它们作为剥离的二进制文件提供[7,8]。但是这样的技术仅仅让逆向模型变得困难，而不能严格防止模型被盗和重用，以及完整性攻击。

近年来，深度学习技术兴起，被用于方方面面，尤其是移动应用中，为生活的便利锦上添花。移动应用中深度学习模型安全涉及到越来越多的领域，有着隐私保护、模型知识产权保护、模型对抗性攻击防御的要求，针对移动应用中深度学习模型的安全问题研究具有极大的理论价值和实际意义。

本文总共分为五个章节：

第一章引言，简单介绍了研究背景，同时简单叙述了研究意义；

第二章研究现状，阐述了目前对于移动应用机器学习模型安全问题的研究现状，介绍了研究相关知识与关键技术；

第三章研究方法设计概述，详细阐述了研究数据集、研究流程设计，分为识别深度学习应用、提取深度学习模型、完整性度量方法三个部分的研究方法与设计方案；

第四章方法实现与结果分析，介绍了实验环境，详细阐述了实现方案与实现细节，展示并分析了研究实验结果；

第五章总结与展望，对本文所完成工作进行了总结，并分析本文研究的优势与不足，并提出下一步的研究与改进方向。

# 第2章 研究现状

随着深度学习在移动应用中的使用越来越流行，移动应用中的深度学习模型安全问题也越来越受到研究的关注。

据统计数据表示，早期的移动应用深度学习采用者都是顶级应用程序。2017年，率先开发出针对于移动设备的深度学习框架的均是顶尖大公司或者独角兽企业。它们更倾向于在自己的产品应用中使用深度学习模型，也有着更为丰富的功能和良好的用户体验。同时，移动深度学习框架越来越受欢迎。尽管完全成熟的深度学习框架（如TensorFlow）由于其发展势头仍在深度学习应用程序中流行，但为移动设备有限资源而设计和优化的深度学习框架越来越受欢迎。

此外，深度学习越来越多地被用作核心构建块。81％的深度学习应用程序使用深度学习来支持其核心功能[5]。也就是说，如果不使用深度学习，这些应用将无法运行，其核心功能无法运作。其中照片美化是最主要的用途。深度学习以其多样化的应用而闻名，这一点已被我们发现的用法所证实，例如表情预测和语音识别。

移动应用中的深度学习模型的保护严重不足。据2019年Xu的工作[5]，只有39.2％的模型被混淆保护，19.2％的模型被加密保护。绝大部分的模型很容易提取，因此容易受到未经授权的重用和攻击者的恶意攻击。

目前对于移动应用中深度学习模型安全问题的研究，能够做到识别和提取，对于其安全问题的分析，未有深入研究；很少有研究在移动平台上进行针对模型完整性的度量分析。

## 2.1 移动应用中深度学习模型识别

在使用深度学习应用的时候，我们往往并不会注意到深度学习在其中所起的作用，只有留心注意并思考时，或可感受到一二。但科学研究不能够只是感性认知，其中Xu的主要成果是分析Android应用中的深度学习支持，识别Android应用中的深度学习使用情况并提取相应的深度学习模型以进行检查，因此可以自动化分析众多深度学习应用。

其中识别方法的实现，是基于大多数深度学习应用程序都是在深度学习框架之上开发的原理。该方法避免困难——寻找特定的代码模式，而是通过确定已知深度学习框架的使用进行判断。识别深度学习模型的核心思想是检测流行框架的使用，该方法存在一定的误报率和漏报率，仍需要手动逆向工程进行检查和分析。

## 2.2 移动应用中机器学习模型提取

Xu的论文中是通过验证框架相应格式是否能被正确解析和文件后缀名来判断是否为模型文件。2020年，Sun Z, Sun R等研究人员对其进行了改进优化，并提出在运行时内存中提取模型的策略：

（1）提取本地未加密模型。混合检测文件后缀名和文件路径关键字，并通过文件大小、apk是否使用深度学习库、文件后缀名是否合理的规则进行过滤，得到本地可能的模型文件；

（2）判断本地加密模型。文件中的高熵值通常是由加密或压缩引起的[10]，使用标准的熵测试可以判断是否为加密模型。据统计实验工作图2-1[9]所示，非加密模型的熵值均在3.5-7.5之间，而加密模型拥有较高的熵值明显接近8。通过文件类型和魔数排除压缩文件，采用7.99作为加密熵阈值，能够有比较好的判断加密模型的效果。

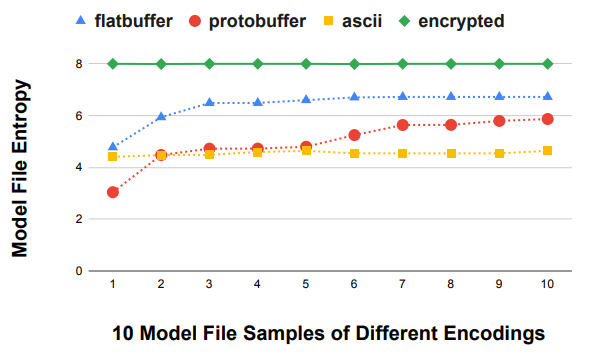


图2-1 4种常见编码的模型文件的熵

（3）运行时在内存中提取模型。设定相应的模型提取策略，默认在加载模型的缓冲区重新分配之前，此时模型加载且已被解密，进行模型提取。除此之外，还有四个替代策略，以适应各种情况，包括在堆中提取、检测模型加载API在其被调用后提取、检测模型解密API在其被调用后提取、在自定义内存分配中提取。

该方法达到了66%的提取率，如图2-2[9]所示。

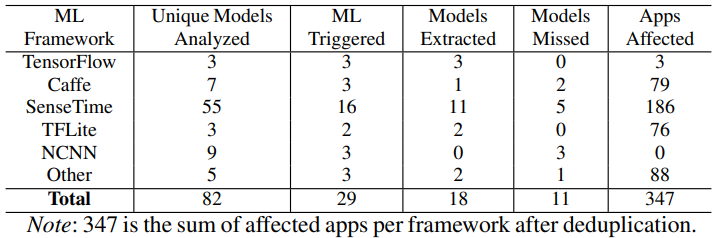


图2-2 模型提取统计

## 2.3 针对模型的完整性攻击

近年来，深度学习等核心技术已被广泛应用于图像处理、语音识别、自动驾驶、垃圾邮件过滤以及人工智能反欺诈等现实任务。深度学习深入日常生活，不仅针对移动应用中的深度学习攻击将造成难以估量的隐私泄露问题，并且利用深度学习赋能安全的使用也越来越多，其环节中的安全问题也逐渐浮出水面——一旦移动应用中用于安全检测相关的深度学习部分被攻击成功，将造成严重的安全问题。

随着人工智能安全研究的进一步深入，深度学习模型安全和隐私问题逐渐引起了广泛关注。Papernot等人将机器学习模型安全需求总结为机器学习CIA模型：机密性（Confidentiality）、完整性（Integrity）和可用性（Availability）[11]。其中，完整性要求模型的学习和预测过程完整不受干扰，输出的预测结果不偏离预期符合正常表现。

CIA三个特性都有可能被所对应的攻击破坏。完整性攻击发生在模型的训练阶段或预测阶段。在训练阶段，最常见的攻击方式是投毒攻击[6]，即攻击者通过篡改训练数据或添加恶意数据来影响模型训练过程，最终降低其在预测阶段的准确性。在预测阶段，最典型的攻击方式是对抗样例攻击，即攻击者通过在测试数据中添加精心构造的微小扰动达到让模型预测出错的目的。模型完整性面临的攻击是对抗攻击，主要分为逃逸攻击和投毒攻击。其中逃逸攻击主要是通过生成对抗样本的方式来逃出模型的预测结果，投毒攻击主要是从数据层面对模型进行干扰[12]。

### 2.3.1 逃逸攻击

逃逸攻击（Evasion Attacks）是指攻击者在不改变目标机器学习系统的情况下，通过构造特定的输入样本以完成欺骗目标系统的攻击。例如，攻击者可以修改一个恶意软件样本的非关键特征，使得它被一个反病毒系统判定为良性样本，从而绕过检测[13]。如图2-3[14]所示，攻击者为熊猫图片增加少量干扰，人类可以轻易地将其判断为熊猫，而深度学习系统会误认为长臂猿。其根本原因在于机器学习模型没有完美地学到判别规则，攻击者就有可能构造对抗样本用以欺骗机器学习系统。

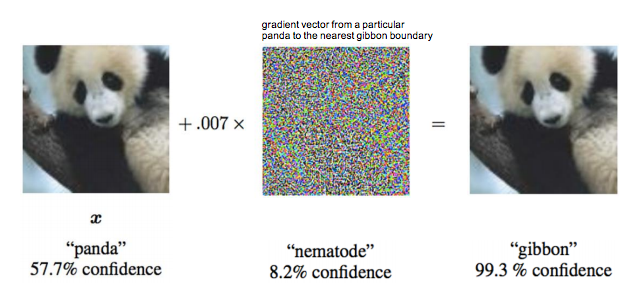


图2-3 在图片中添加扰动导致深度学习系统错误识别实例

基于机器学习的逃逸攻击可以分为白盒攻击和黑盒攻击。白盒攻击需要获取模型算法和参数，获得对抗样本；黑盒攻击只通过知道模型输入输出并观察模型输出变化生成对抗样本。

目前逃逸攻击的防御主要从输入、模型网络、外部防御三个方向入手，防御措施有对抗训练、梯度掩码、随机化、去噪等，各有相应优点、缺陷和适用范围。

### 2.3.2 投毒攻击

投毒攻击(Data Poisoning Attack)也称数据中毒攻击，是将异常数据加入到模型的训练数据中，导致模型出现错误的输出结果。

此类攻击常常出现在系统获取数据不断更新模型的过程中，例如Google垃圾邮件分类器就曾发生过数据投毒攻击案例。2017年11月底至2018年初，Gmail遭遇了至少4次大规模恶意攻击，试图让分类器偏斜，如图2-4 [15]所示。攻击者试图通过将大量垃圾邮件提交为非垃圾邮件来污染训练数据，让模型对输入的分类发生偏移，从而降低了模型的准确率。



图2-4 2017年11月底至2018年初Gmail遭遇了4次大规模恶意投毒攻击

投毒攻击的一般防御方法有训练数据过滤、回归分析、集成分析。在具体生产过程中措施有，构建标准数据集使模型能够准确预测后才能投入生产，确保评估反馈的真实性之后再与其他特征相结合。

## 2.4 Android逆向

研究Android应用需要有一定的Android逆向基础。本节将简单介绍研究中所需要了解的Android逆向基础知识。

Android应用程序的文件是apk文件，apk文件可以看成是一个zip压缩文件，包括classes.dex（Dalvik字节码，编译后的java源码），resources.arsc（资源文件），AndroidManifest.xml等，apk反编译后的典型文件结构如图2-5[16]所示。

Apk反编译工具Apktool将apk文件解压缩，并解码其内容。对于资源文件，不需要做很多事情；对于Dalvik字节码claasses.dex（DEX即Dalvik excutable file，是Google专为Dalvik设计的字节码可执行文件），会被反编译为smali代码，Apktool会将输出文件放置到创建的与apk文件名称相同的文件夹中。文件夹中包含XML资源文件，AndroidManifest.xml文件，源代码等。XML资源文件和AndroidManifest.xml文件易于阅读，通常非常接近原始格式。反汇编的smali代码位于smali文件夹中，通常，一个smali文件包含一个Java类的代码。

### 2.4.1 Android运行机制

Android底层内核空间以Linux Kernel作为基石，上层用户空间由Native系统库、虚拟机运行环境、框架层组成，通过系统调用（Syscall）连通系统的内核空间与用户空间。用户空间主要采用C++和Java代码编写，通过JNI技术打通用户空间的Java层和Native层（C++/C），从而连通整个系统，如图2-6所示。

Java层通过Java虚拟机（目前有Dalvik、ART等Java虚拟机种类）执行Java层可执行文件，执行代码的过程其实就是Java虚拟机在解析字节码。

Native层主要是以C++/C语言编写生成的so（shared object）文件的形式存在apk的lib库中，在使用System方法进行加载进而使用。

因此，Java有两种方法，Java方法与本地方法。Java方法由Java语言编写，本地方法由其他语言，例如C、C++、汇编编写，编译成处理器相关的机器代码，保存在动态链接库中。Java程序调用本地方法时，首先使用System.loadLibrary将包含本地方法的动态链接库加载进内存，当Java程序需要调用本地方法时，虚拟机在加载的动态链接库中定位并链接本地方法以执行。使用registerNative方法注册类中本地方法可以让程序主动将本地方法链接到调用方，不需要虚拟机链接定位的步骤。

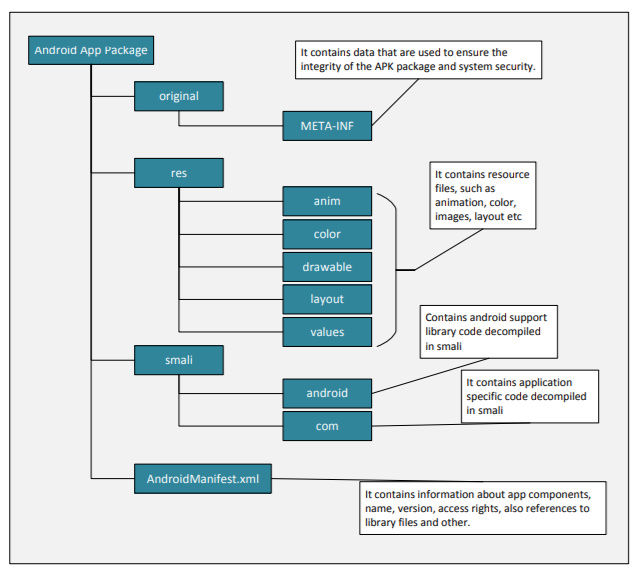


图2-5 apk文件结构

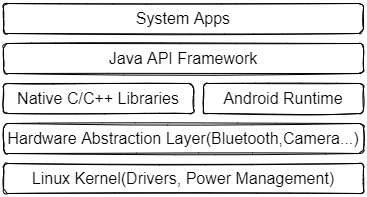


图2-6 Android系统架构层

### 2.4.2 Android使用深度学习

在智能手机上使用DL，可以将模型处理在云端完成，也可以仅部署在智能手机上执行。相对于云端DL，设备上DL有更好的隐私保护，对网络波动造成的影响有一定的抵御能力。目前设备上DL使用更加广泛，移动应用程序开发人员往往是使用云服务器事先进行离线培训建模，在应用安装时，将训练好的模型打包进应用安装包进行部署，在应用运行时，移动应用通过调用DL框架对训练好的模型进行推理使用。

Android中使用深度学习的具体实现方式大多是使用导入相应本地lib库。

深度学习模型第三方供应商会给使用方提供相应的模型API接口，模型参数相关文件，模型使用者只需要在使用前加载模型供应商所提供的lib文件即可。加载深度学习动态链接库文件（DL lib）的方式有以下几种：

（1）Java层：使用System.loadLibrary(nativelib);方式加载lib文件，加载lib文件后会执行lib文件中的JNI\_OnLoad函数，并在Java层声明对应要调用的native层lib导出函数；

（2）Native层：使用dlopen或mmap函数加载lib文件；Android系统中，对链接库进行加载的程序是/system/bin/linker，linker加载动态链接库的函数是dlopen，Java层的System.loadLibrary也同样基于dlopen函数；

（3）动态注册：出于安全性考虑，现在更多地使用动态注册绑定native函数，通过调用RegisterNatives函数实现。

加载lib在使用时也有相应的加固保护技术增强其安全性，例如Alipay支付宝使用了程序保护措施，将加载lib部分的代码预先编译为字节码，通过这种方式在加载过程中不会调用到load、dlopen等相关加载函数，加大动态调试和插桩的难度。

## 2.5 环境及工具使用

（1）Apktool

Apktool[17]是Android应用程序逆向工程工具，它可以将资源解码为近乎原始的形式，并可以在进行一些修改后重新构建它们，主要用于本地化地添加一些功能或自定义平台支持，以及分析应用程序等。

（2）Android测试机root环境

对Android应用进行分析的环境使用真机环境最好，x86处理器下的Android虚拟机与ARM处理器架构下真机Android有一定区别，市场大部分apk无法直接在x86虚拟机上运行。

对Android应用进行调试需要满足root环境，勾选开发者选项允许调试。本研究中使用的Android环境为Nexus 5X作为测试机，操作系统是Android8.0版本，使用twrp[18]、Magisk等工具进行root环境搭建。

（3）Xposed

Xposed[19]是一个运行于Android平台的钩子模块框架。Xposed通过替换Android操作系统关键文件/system/bin/app\_process，扩展修改后会将XposedBridge.jar加载到运行时环境，在Java虚拟机启动之前执行一些操作，并允许通过Xposed模块自定义代码更改调用Java函数时的行为。

例如，在Xposed中安装激活Xdebuggable模块可以改变Android应用是否可调式的标志位，能够更方便地调试移动应用程序。

（4）Frida插桩工具

随着Android平台的蓬勃发展，Frida[20]插桩工具应时而出，为开发人员、逆向工程人员和安全研究人员等群体，提供了方便实用的动态命令工具包。通过编写Javascript注入脚本，可以没有源代码的情况下实现hook函数、监视API、跟踪应用程序代码等功能，并适用于Windows、macOS、Linux、Android等多种平台，免费开源，通过各种大规模测试并且一直在维护更新。

Frida是一款主要在电脑上操作的工具，这决定了其具有高并发、多联通、自动化等特性。高并发，可同时操作多台手机，调用多个手机上的多个应用中的算法；多联通，电脑与手机相互连通，移动端无法处理的分析部分可以在电脑端进行处理；自动化，移动端电脑端互相协同，实现横跨平台协同自动化。

Frida作为时下流行的代码插桩工具，可以向移动应用中注入代码，以进行动态调试以及其他相关测试，针对同时支持Native层和Java层。

本研究中由逆向工程先行，进行Android程序中模型相关使用的静态分析和动态调试，进而使用Frida插桩工具向应用注入相关分析的代码，有着手工逆向工程无法做到的半自动化。

## 2.6 本章小结

本章阐述了目前对于移动应用机器学习模型安全问题的研究现状，包括对使用深度学习的移动应用的识别方法、移动应用中深度学习模型的提取方法，以及针对深度学习模型的完整性攻击，同时还介绍了研究所需要的相关知识与关键技术，主要是Android运行机制以及移动端如何使用深度学习的Android逆向基础，最后简单介绍了研究中用到的相关环境的配置与使用工具。本章内容主要对于研究方法设计的前期准备。

# 第3章 研究方法设计概述

## 3.1 流程概述

对移动应用中深度学习模型进行完整性度量分析的工作流程如图3-1所示，以半自动方式运行。

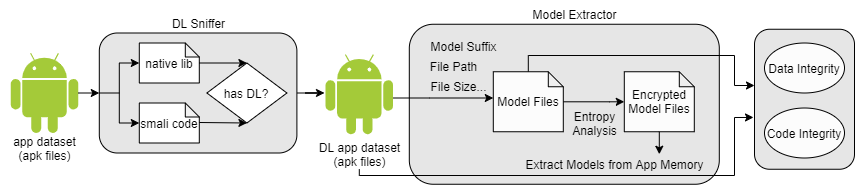


图3-1 工作流程

工作流程的第一步是在一组给定的Android应用作为输入识别其中的深度学习应用。这是通过名为DL Sniffer的模块来实现，检测流行的深度学习框架的使用，而不是直接查找深度学习的使用，采用了Xu提出的识别算法，更全面地进行过滤选择。识别深度学习应用程序后，会对这些应用程序包含的深度学习相关文件进行简单的分析。

第二步是从识别出深度学习的应用程序中提取深度学习模型。该步骤中的提取模块名为Model Extractor，结合了Xu与Sun Z, Sun R等研究人员提出的方法中的思想，动态方法与静态方法结合。首先对本地可能是深度学习模型的文件进行检测，判断是否为加密模型，之后对于加密模型和未检测出本地模型的样本采取使用Frida工具在运行时提取模型的方式。

最后一步，是对移动应用中的深度学习模型进行完整性度量和分析。该步骤由Integrity Analyzer模块实现，分为两个方面：模型数据完整性，包括模型参数完整性、模型格式完整性；模型使用（代码）完整性。

在模型提取和模型分析部分，遇到的挑战是模型使用的不同框架以及不同格式。尽管有开发人员投入精力集成不同的模型格式，设计标准化模型格式，DL框架的生态系统依旧支离破碎。因此，在研究DL模型时将重心主要放在了有据可查的开源框架上。

接下来将分别讨论DL Sniffer、Model Extractor、Integrity Analyzer的更多细节。

## 3.2 识别深度学习应用

作为分析流程的第一步，DL Sniffer识别Android应用中的深度学习支持的思路主要是，识别其中的lib文件特殊字段以及相关API调用。

通过感性感知归纳深度学习应用的共同点，以此去分辨深度学习应用，相对来说是困难、费时费力、不精确且难以自动化高效完成的。故而将抽象的识别使用了深度学习的应用的问题，转换为识别主流深度学习框架、识别深度学习调用方法等特征，避免困难的研究——寻找特定的代码模式，通过确定已知DL框架的使用进行判断，是更加精妙有效的方法。

其中识别方法的实现，基于大多数深度学习应用程序都是在深度学习框架之上开发的原理。由于开发DL框架需要大量的工作和成本，因此大多数应用程序开发人员更倾向于使用可靠的现有框架。这使得通过识别DL框架来识别DL应用可行可靠。

DL Sniffer识别示意图如图3-2所示。具体识别策略如下：

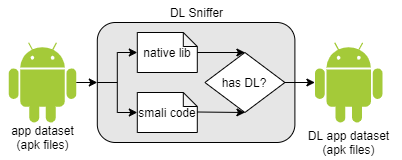


图3-2 DL Sniffer识别DL应用

（1）对于使用本地库原生Native API的应用程序，提取反编译后lib文件夹中的.so文件的rodata字段（read-only，只读数据字段，存放常量），查找其中对应框架的特定字符串，不同框架被使用时该段会有相应不同的字符串标识。例如，TensorFlow在其相关lib文件的rodata节存在字符串“TF\_AllocateTensor”；

（2）对于仅使用Java API的应用程序，获取反编译后的smali文件，查找class、methods字段，检查是否含有相应框架的特定API。例如，“MultiLayerConfiguration”API被几乎所有的DeepLearning4J框架使用；

（3）对于其他未使用框架的使用，则在rodata段搜索深度学习相关的特定字符串，例如“neural network”、“convolution”、“lstm”等字符串。

该方法存在一定的误报率和漏报率，仍需要手动逆向工程进行检查和分析。此外对于进行了加固保护的样本，也需要手工进行逆向工程分析处理。

## 3.3 提取深度学习模型

第二步由Model Extractor提取已经被检测为DL应用的样本中的DL模型。Model Extractor采用静态方法与动态方法相结合的方式，提取思路是寻找相应DL框架的模型特征，包括文件信息特征与模型内容特征，依据其在本地静态提取，在运行时内存中动态提取，最后对提取到的模型进行验证。

Model Extractor的工作流程如图3-3所示。

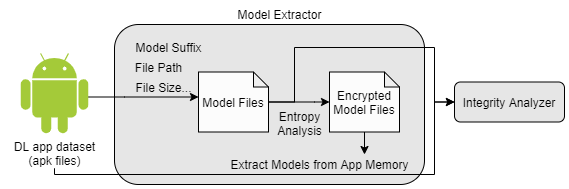


图3-3 Model Extractor提取DL模型

**静态方法**：由于各种DL框架模型格式不同，Model Extractor会使用DL框架相应的模型后缀名与文件路径中是否含有相应关键词混合检测，初步判断每一个模型文件。但如果仅这样判断，会有一定的误报率，所以接下来Model Extractor将过滤掉文件大小、文件后缀名不合理的文件。

检查模型可解析性、文件大小等特征，针对受支持框架采用相应的验证器，尝试验证每个模型文件，获取本地DL模型文件与可能被加密混淆的模型文件。

之后计算模型文件的标准熵，以判断模型文件是否被加密混淆处理。文件中的高熵值通常是由加密或压缩引起的，使用标准的熵测试可以判断是否为加密模型。非加密模型的熵值均在3.5-7.5之间，而加密模型拥有较高的熵值明显接近8，故采用7.99作为加密模型的熵阈值，有比较好的判断加密模型的效果。对于存在加密模型与未被检测出本地模型文件的样本，Model Extractor将会对其进行逆向工程，尝试使用动态方法提取DL模型。

**动态方法**：在移动应用运行时，加密模型会在内存中被解密成明文，所以在模型未受到保护的情况下，内存分配释放函数调用之间大概率会有模型明文存在，通过使用插桩工具Frida，在合适的时机能够得到解密后的模型。

Model Extractor通过一定的代码检测策略来判断模型提取时机。运行时提取模型最合适的时机是在模型内存缓冲区释放之前，因为相应内存缓冲区释放时，内存缓冲区已经完全加载和解密模型；相较于内存分配API（例如malloc、calloc），内存释放API（例如free）调用数量有限并且相对容易进行插桩。

对内存释放API进行插桩跟踪的方法可能会导致应用程序运行速度显著下降，为了减小该问题的影响，优化执行效率与可用性，Model Extractor会判断DL库加载时机，仅在其加载后对于大于最小模型大小的缓冲区进行插桩跟踪。为了获取内存缓冲区大小，还需要检测内存分配API，记录其分配大小。

当移动应用程序不适用本地DL库，或使用自定义的内存分配释放方法，或动态分析使用时不释放模型缓冲区的情况下，这种检测策略可能会失败。在这些情况下需要检测特别策略或手动逆向工程。

在恰当的时机捕获可能包含DL模型的内存缓冲区之后，Model Extractor会基于模型表示识别内存模型，即检测不同格式的模型编码格式。各个模型格式和表示形式都有相应的内容特征与编码签名，在实验研究过程中，发现例如TFLite模型，通常包含“TFL2”、“TFL3”作为文件头魔数字段，Protobuf也有相应的编码规则。

除此之外，模型大小也是重要的内存模型识别判断因素，与加密模型大小相同的缓冲区包含解密模型的可能性极高，因为在实现解密过程时，开发人员几乎总是分配与加密内容大小相同的缓冲区保存解密内容，这样既无需精确计算揭秘缓冲区大小又相对安全（解密内容总是短于加密内容）。

在模型提取工作完成之后，进行模型验证，与静态方法同样使用开源的模型解析器验证模型的可解析性，并验证是否与加密模型、缓冲区大小或其他模型相关信息匹配。

## 3.4 完整性度量方法

最后由Integrity Analyzer对DL应用样本的DL模型进行完整性度量分析。在模型存储和模型使用两过程方面对模型完整性进行度量，可将其分为数据完整性与代码完整性。对于模型的数据完整性，度量其存储信息，模型是否明文是否可篡改，加载时是否有校验模型文件完整性；对于模型的代码完整性，通过跟踪DL本地库中的相关函数调用，分析其控制流以进行度量。

Integrity Analyzer的工作流程如图3-4所示。

在模型存储过程中，若模型明文存储，其格式可解析可篡改，并且在移动应用程序加载时又未校验本地文件完整性，攻击者即可通过篡改本地模型改变应用程序中使用DL进行推理的结果，可以达到绕过某些安全检测的效果。Integrity Analyzer会跟踪DL模型加载相关函数，来判断其是否存在被篡改的可能性。

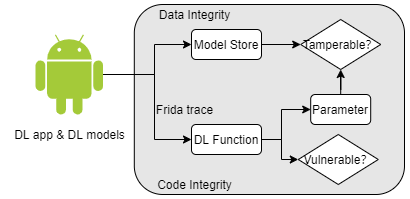


图3-4 Integrity Analyzer完整性度量

在模型使用过程中，跟踪DL本地库导出的DL模型相关函数，分析模型使用控制流，检测是否存在使用不当的地方，以致模型使用相关输入或者参数可控，存在完整性缺陷。

## 3.5 本章小结

本章主要是对研究目的与研究数据集选取，以及移动应用中深度学习模型的完整性度量方法的设计方案的详细阐述。设计方案主要分为三个部分，深度学习应用的识别，深度学习模型的提取，以及深度学习模型的完整性度量。

对于识别深度学习模型，主要对识别方法进行了详细阐述，介绍了DL\_Sniffer的工作流程，解释了方法的核心思想和选取原因。

对于提取深度学习模型，介绍了模型会有的本地格式以及相关特征，并根据其在已有方法的基础上对静态方法进行了改进，结合动态方法，详细阐述了移动应用中深度学习模型的提取方法。

对于深度学习模型完整性度量方法，主要介绍了度量采取的影响因素，与相应的流程设计。

本章内容是全文设计的主体部分，是对于本研究方法设计的详细阐述与解释。在下一章中，主要叙述了方法的实现细节，并分析了得出的数据结果。

# 第4章 方法实现与结果分析

## 4.1 实验环境与数据集

本次设计研究的环境如表4-1所示：

表4-1 实验运行环境

|  |  |  |  |
| --- | --- | --- | --- |
|  | 品牌型号 | 操作系统 | 内存 |
| 实验环境 | Google Nexus 5X | Android8.0 | 2G |
| 运行环境 | Dell | Windows10/WSL1 Ubuntu16.04 | 16G |

实验环境选择了最为经典、使用量大、兼容性好的Android8.0系统版本，采用了Google Nexus 5X原生安卓系统，原生Android系统更加可靠，实验效果不会因为其他第三方厂商自主开发的操作系统或某些配置设置出现不可预料的结果；构建的分析过程工具的运行环境为本机Linux子系统，由于数据集较大，且有较多文件读写过程，使用子系统能够同时具备Linux实验环境同时有相对较好的文件读写效率。

研究实验主要使用python语言编写，环境配置需求如图4-1所示：

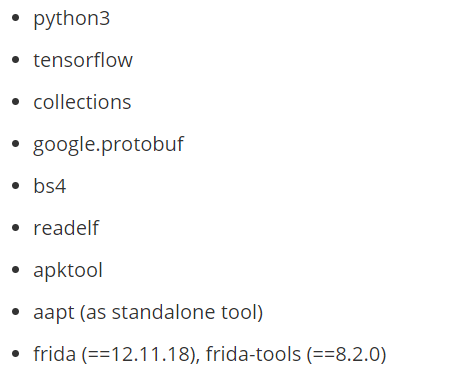


图4-1 研究实验环境配置需求

**范围** 本文将分析重点放在Android应用程序上，因为Android代表了智能手机出货量和使用量的主要部分，因​​此可以很好地代表整个智能手机应用程序群体。

**数据集** Google Play商店（通过apkpure网站下载）和360手机助手应用市场的各个类别中综合排名靠前的免费应用程序。对于每个应用，下载其apk文件并从相应网页抓取其元信息（例如，应用说明和用户评分）进行分析。实验样本数据如图4-2所示。



图4-2 来自360手机应用助手的实验样本数据集

## 4.2 方法实现

根据设计方法，研究过程将分为识别深度学习应用、提取深度学习模型、完整性度量分析三个部分，将在本章小节分别具体讨论。

### 4.2.1 识别深度学习应用

识别DL应用的代码流程如图4-3所示。

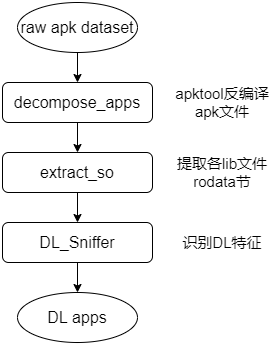


图4-3 DL Sniffer识别DL应用代码实现

（1）运行decompose\_apps\_apktool.py代码，使用apktool工具反编译数据集中的apk文件。得到各apk文件反编译后的文件夹后，对不同文件采取不同的对应分析：

AndroidManifest.xml：根据文件内容提取程序运行相关信息，包括进程名、启动时的Activity类等；

lib文件夹：lib文件夹中存放的是Android应用运行中使用到的动态链接库文件，是由C++/C语言生成的so文件，其中各个lib文件的rodata节将在下一步中被提取；

smali文件夹：smali文件夹中存放的是dex字节码反编译得到的smali文件，对其将进行DL相关API的查找；

资源文件：本地可能的模型文件往往存放在assets文件夹中。

（2）运行extract\_so.py代码，使用readelf工具提取出apk样本中所有lib文件的rodata节。

（3）运行DL\_Sniffer.py代码，DL\_Sniffer在预先提取的各应用lib文件rodata段中，查找是否存在各个DL框架的预定义特征字符串。如果存在相应字符串标识，则该lib文件被判定为为DL本地库。包含DL本地库的应用被认为是使用了深度学习的应用，即DL应用，接下来会根据是否存在DL本地库生成识别出的DL应用名的文件DL\_apps.txt。各DL框架以及预定义的特征字符串如图4-4所示。

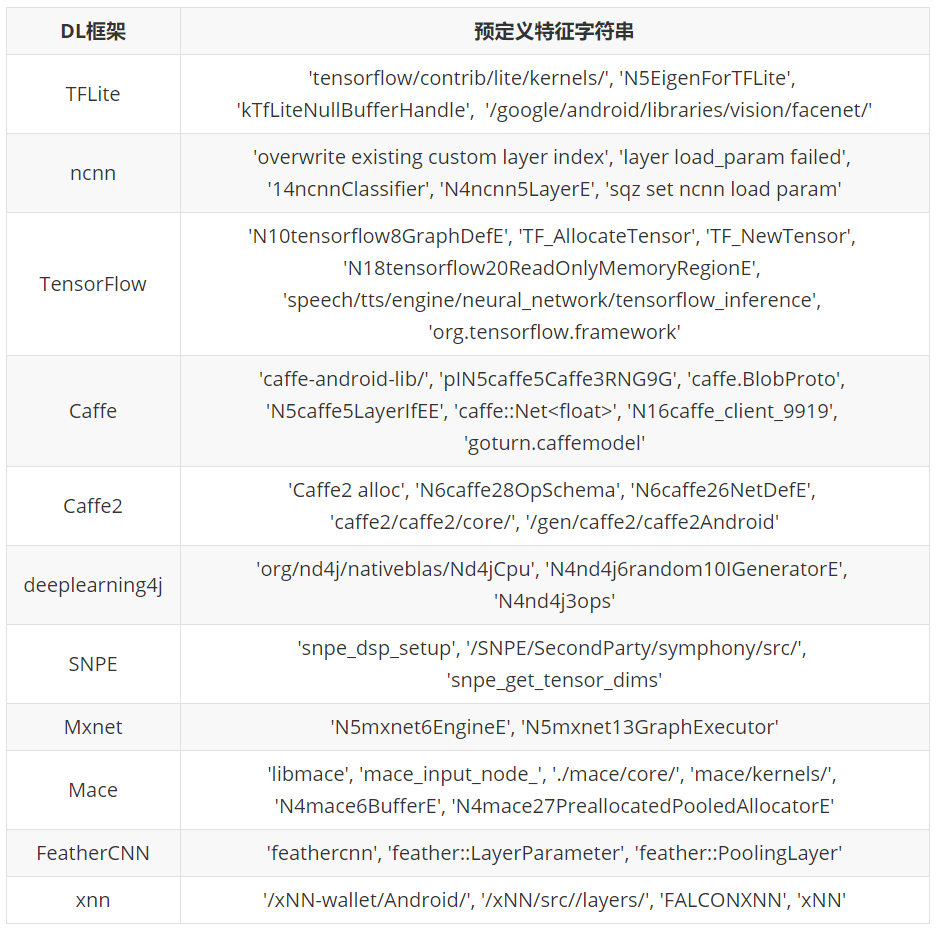


图4-4 受支持DL框架的预定义特征字符串

### 4.2.2 提取深度学习模型

提取DL模型的代码流程如图4-5所示。

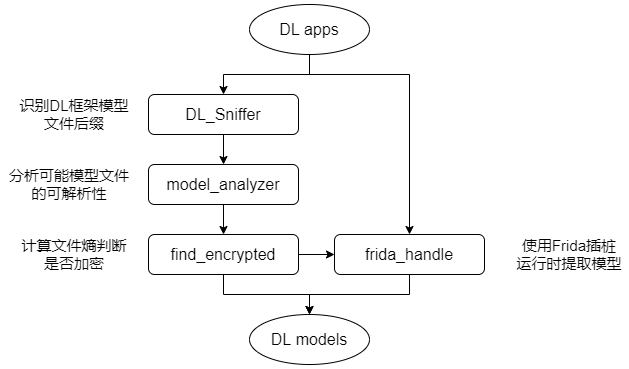


图4-5 Model Extractor提取DL模型代码实现

（1）运行DL\_Sniffer.py代码，根据应用程序检测出的对应DL框架类型，查找本地可能的DL模型文件，生成包含DL模型文件路径、文件大小等信息的文件DL\_model\_path.txt。具体探查判断DL模型的detector策略如图4-6所示。

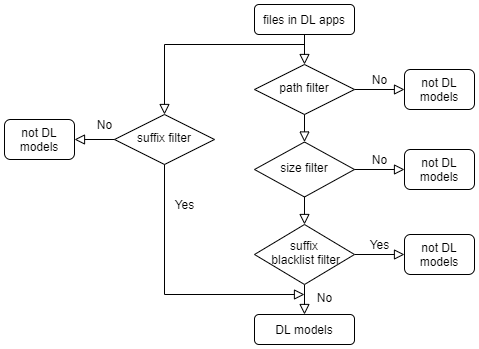


图4-6 detector判断策略

对于使用相应DL框架专用模型格式后缀的模型，直接判断为对应DL模型；对于路径中包含有相应关键词的文件，也可能是DL模型，但还需要判断其文件大小是否符合一个模型该有的文件大小，文件类型后缀是否是不可能是DL模型的后缀（例如，jpg，png，doc等），以此进行过滤，降低误判的可能。

（2）运行model\_analyzer.py分析提取模型的可解析性。如果能够解析，则会进一步对受支持框架进行分析。

（3）运行find\_encrypted\_model.py代码，计算模型文件的熵导出至entropy.txt，得到经过加密混淆的模型文件名存至encrypted\_DL\_models.txt。

计算文件标准熵的代码实现具体步骤是，统计文件中各字符字节数，根据标准熵公式，累加每种字符出现频率与频率的log的成绩，得到文件的熵值。

（4）运行frida\_handle.py代码，该部分代码使用了Frida工具对移动应用进行插桩。

首先判断DL相关lib文件的加载时机，以进行相关hook操作：

1）枚举已加载模块modules，判断目标DL lib是否已经被加载；

2）hook加载lib文件的Java层函数System.loadLibrary、System.load，判断参数是否为DL lib；

3）hook Native层函数dlopen，linker在加载动态库时调用的函数时dlopen，Java层的loadLibrary函数同样基于dlopen函数；

4）hook函数RegisterNative，RegisterNative是Native函数进行动态注册的函数。判断函数模块module是否为DL lib，可以判断出lib是否被加载。

检测到lib文件加载后，对lib文件进行相应hook操作：

5）hook free malloc等内存分配相关函数，记录内存缓冲区分配信息。

### 4.2.3 完整性度量

进行完整性度量的代码流程如图4-7所示。

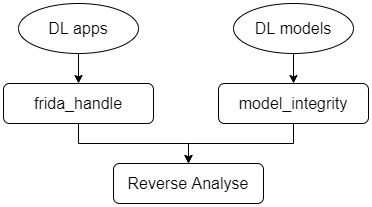


图4-7 完整性度量代码实现

（1）运行frida\_handle.py代码，hook之前检测出来的DL应用对应的lib，hook相关DL使用函数，得到DL函数调用情况。Hook的内容有exports导出函数，RegisterNative进行注册的函数，跟踪DL相关函数被调用情况，记录调用顺序、调用关系，以及函数参数等；对于DL模型加载函数，判断是否存在完整性校验。

（2）运行model\_integrity.py代码，根据之前步骤中检测出的本地模型是否加密混淆、是否可解析，结合frida\_handle过程中得到的加载过程是否存在完整性校验进行数据完整性度量。

（3）手工逆向查缺补漏深入具体分析。

## 4.3 结果分析

### 4.3.1 深度学习应用程序

对于样本数据集，执行识别后得到的数据如图4-8所示。

如图所示，样本中的DL应用程序总数为129，占样本总数的13.2%，2019年Xu的研究中，在16,500个Android应用程序中确定了211个DL应用，占所有应用程序的1.3％；2020年Sun Z, Sun R等人的研究，在46,753个Android应用程序中确定了1,468个使用设备上ML的应用，占整个应用集合的3.14%。相比之下，2021年本研究的数据有大幅增长。

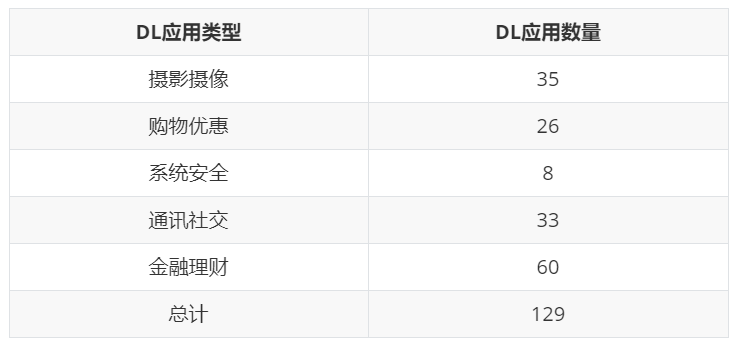


图4-8 DL应用

这样的结果有以下原因：一是近年来随着深度学习在移动平台的发展，移动应用中使用DL能够实现更为丰富多样化的功能，DL在移动应用程序中的使用越来越多，应用程序不断推陈出新，增加DL功能和新开发DL的数量都在增长；二是综合排名越靠前的应用程序中，DL应用程序占比越高，原因是综合实力强的大公司更有能力开发优秀的DL应用，并且使用DL的应用程序功能更加高级，更受用户欢迎。

以上实验结果表明，如今有大量的智能手机在运行DL应用程序，DL应用程序越来越青睐，同时DL也在推进移动应用的发展。

深度学习使应用程序的功能更加丰富，被广泛地用于应用程序安全检测、身份认证功能，被大量日常软件作为核心功能使用，这也意味着DL性能和安全性将会显著影响智能手机上移动应用的用户体验，即应用使用效果及其是否安全可靠。

根据实验数据，使用DL的应用程序中，金融理财、摄影摄像和通讯社交类别相对较多，对应身份识别认证与图像美化的DL功能被广泛地使用。身份认证包括人脸识别、身份证识别、银行卡识别等涉及到个人信息隐私安全，在很多应用中被作为身份识别、实名认证模块使用；图像美化包括人脸识别、动作识别等，多数摄像、美图应用都进行了使用，同时也作为非核心功能被广泛应用在社交软件中。其中摄影摄像等类别更多的是使用DL构建实现应用程序的核心功能，而金融理财、系统安全等类别的应用使用DL更多是在非核心功能中。安全认证的DL功能被DL应用的广泛使用，也意味着DL在应用中安全使用的重要性。

### 4.3.2 深度学习框架

大多数开发人员选择使用DL框架进行移动应用中DL功能的开发，在研究实验过程中，检测了多种流行DL框架的使用情况。

大多数流行的DL框架通过Java或提供C++交叉编译库支持Android平台，大多是开源的，另一部分作为二进制SDK提供，或仅有提供商自己的产品访问。这些DL框架中的大多数使用定制的格式存储和表示模型文件。

由于培训模型在数据和计算方面都相当复杂，因此移动应用开发人员主要依靠云服务器在应用程序部署之前进行离线培训建模。应用程序运行时，调用DL框架对预先训练好的模型进行推理使用。此外，在检测出的DL应用中，专门为移动应用程序设计，没有训练支持后端的轻量化的DL框架越来越受欢迎，例如ncnn，MACE等。这些轻量化DL框架无法训练DL模型，但可以使用通过TensorFlow、Caffe等其他框架预先训练的模型进行预测。

如图4-9所示，最受欢迎、使用最多的DL框架是Caffe、ncnn、TFLite。这些框架的有一些共同点：它们都是由大公司开发的（TensorFlow、TFLite-Google，ncnn-腾讯，Caffe-Berkeley，Caffe2-Facebook，xNN-Alibaba等），大多开源，其中大多针对智能手机平台进行了优化。



图4-9 DL框架

DL框架的广泛使用意味着，DL框架本身的安全性以及使用DL框架的安全问题也会影响到移动应用的用户体验。

### 4.3.3 深度学习模型

DL应用中的DL模型类型虽然不及应用类型繁多，但也具备相当丰富的功能种类，其中最常见的是人脸识别、手写识别、语音识别、生物识别以及ID卡、银行卡识别等DL模型应用。图4-10统计了深度学习模型类别、数量及代表模型文件。

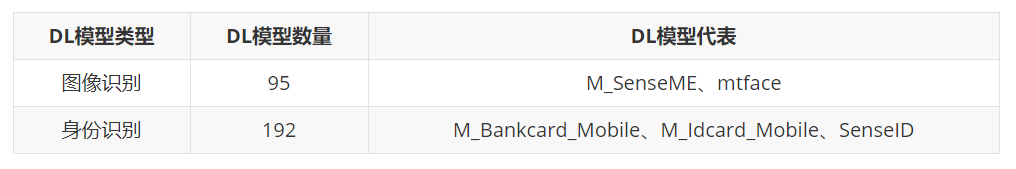


图4-10 DL模型

使用的最多的模型类型是身份识别和图像识别，其中SenseTime的DL模型备受欢迎，图像识别和ID卡识别相关模型被用于个应用中，主要实现人脸识别、ID卡识别等功能。文本识别、语音识别的DL模型也被经常使用。

图4-11展示了各类别DL应用中检测出的DL模型数量。可以看到DL模型数量远高于DL应用数量，一个DL功能有时需要多个DL模型实现。

一些DL应用会使用具有不同功能的多个DL模型。银行、借贷等金融理财相关应用中，往往都会有人脸识别、银行卡识别等DL模型；社交软件中检测出的模型数量最高，存在用于人脸识别、图像美化、银行卡识别、身份证识别等功能的DL模型。

在移动应用中使用DL模型已经成为流行趋势，DL模型的多样性也决定了深度学习在移动应用中广泛的使用，在这种趋势之下，保护模型安全对于移动应用的长久发展更是关键。



图4-11 各类型DL应用检测得到的DL模型数量

### 4.3.4 完整性度量分析

对模型的保护措施主要有混淆、加密、使用专用模型格式。混淆通过某些算法（例如字节异或）使有意义的文本变得无意义难以看懂，来防止攻击者深入了解模型结构，这种方法通常浅显且容易被恢复成原文。加密通过密码算法以避免攻击者获取模型结构和参数，但往往带来不可避免的开销。使用专用模型格式，需要攻击者进行相应耗时耗力分析，但开发专有模型格式，同样需要相当的耗费。

研究实验得到的模型保护结果如图4-12所示，有41.3%的模型使用了加密或者混淆的模型保护措施。在Xu的研究中，120个DL模型中有47（39.2%）个模型被混淆，23（19.2%）个模型被加密（加密模型包括在混淆模型中）；在Sun Z, Sun R等人的研究中，只有59%的ML应用程序对其模型进行了保护（包括运行时保护等）。其中涉及到钱财的金融理财类DL应用中的加密模型比例最高。可以看到，模型保护问题逐渐得到开发人员的重视，但还远远不够，没有任何保护措施的模型仍是大多数。

在本文提取DL模型研究实验中，本地明文模型居多，这意味着模型可以被轻易地从应用程序安装包或安装目录中提取，使得作为知识产权的模型文件被窃取重用，同时也使针对模型的对抗性攻击可以通过对明文模型的分析变得更有针对性更加轻易。

本地模型提取以外，能够在运行时内存中动态提取的模型接近半数。提取失败的原因除了在程序中进行了模型保护措施以外，还存在应用运行崩溃等系统问题。大多数对模型进行了加密混淆保护的应用程序，会在内存中解密模型，这意味着模型在内存中完全不受保护，一个掌握逆向工程基础知识的攻击者可以轻松提取。

实验过程中，异或混淆的模型文件的恢复十分轻易就可以判断做到，加密模型也可以通过动态提取的方式得到明文模型，这意味着对于移动应用中模型的安全保护技术亟待发展提高。



图4-12 各类DL应用中的加密模型数量

·使用量更大的DL框架的DL模型上更多地采用了模型保护。被广泛使用的DL模型更需要也更有能力进行模型保护。一些DL框架提供了模型保护机制，例如Mace支持开发人员将模型转化为代码。

·模型供应商相对更重视模型保护。模型泄露会对移动应用程序开发者造成的经济损失更多是竞争损失，对模型供应商则使造成极大的直接损失，所以模型供应商往往更重视并且更需要保护模型以维护其知识产权。例如SenseTime商汤科技在提供模型的同时，提供了模型加密功能的库，对于模型本身也进行了相应的加密混淆，采用了自己开发独有的模型格式。

·大部分DL模型被明文存储在本地，并且在加载时没有进行校验。在模型使用过程中，部分使用了保护措施，如SenseTime提供的DL支持是将模型存储在DL本地库中并进行混淆，在加载的模型的时候进行了校验。

## 4.4 本章小结

本章主要是方法实现的具体细节叙述和实验结果的展示与分析，介绍了实验进行的环境与配置需求，详述了方法实现的代码过程，根据实现得到的数据结果，分别对于研究中的深度学习应用程序、深度学习框架与深度学习模型进行了相应的分析与比较，对移动应用中深度学习模型的完整性安全性问题进行了详述。

通过分析可知，移动应用开发人员在使用深度学习支持的过程中，对其完整性安全问题认识不足，没有足够的模型保护意识；同时对于移动应用中深度学习模型的保护技术和方法也亟待得到更好的发展。

# 第5章 总结和展望

## 5.1 工作总结

本研究对在移动应用使用深度学习模型的完整性安全问题进行了探索，构建了静态与动态结合的工具对市场级的Android应用程序进行了半自动化挖掘和分析。通过对当前已有的移动应用上深度学习模型的安全问题分析的学习研究，以及对使用深度学习模型的应用例子的逆向工程分析，得出了有趣而有价值的发现。研究表明，深度学习在受欢迎的应用程序中扮演着至关重要的角色，移动平台中深度学习模型的完整性安全不可或缺，值得移动应用开发人员与模型供应商的重视。

本文完成的主要工作有：

（1）阅读相关文献，简单描述了移动应用中深度学习模型的应用现状与初步发展的研究现状，同时简述了本课题的研究意义；

（2）设计并实现了识别深度学习应用、提取深度学习模型、完整性度量分析的方法，构建了静态动态结合的半自动化工具，相比前人研究更为完整，效果更好；

（3）提出了在移动应用中使用Frida插桩工具分析深度学习模型的方法，提供了在移动应用中使用Frida的应用实例；

（4）提出重视移动应用中深度学习模型的完整性安全问题，给移动应用深度学习开发人员、深度学习模型供应商一定的启示；同时也说明了Android平台自身安全保护相对匮乏，呼吁Android平台朝着更加安全的方向发展。

## 5.2 研究展望

如今移动平台的兴起与发展势不可挡，使用深度学习实现丰富功能的移动应用渗入到我们生活的方方面面，其面临的安全问题也需要得到相应的重视。本文设计实现了移动应用中深度学习模型的完整性度量方法，但仍然存在巨大的提升空间：

（1）尽管做到了半自动化处理分析，仍需要大量的人工努力分析、排查错误并验证结果，方法存在着一定的误差，有待进一步研究；

（2）相较于前人的研究工作，使用Frida插桩工具加强了动态分析部分，但是动态分析的可控性较差，并且具有相当的运算复杂度，实际使用需要耗费一定的时间和精力，有待进一步研究与优化；

（3）仅分析了深度学习在Android平台的使用情况，尽管Android可以很好地代表移动生态系统，但扩展到其他平台例如IOS等，可能也会有一定的发现；

（4）对新兴的深度学习框架没有进行分析，有待举一反三对更多的深度学习框架与模型进行研究；

（5）计划使用模糊测试对移动应用中的深度学习模型使用部分进行代码完整性度量，后续将继续进行相关研究。

# 致谢

白云苍狗，大学四年进入了尾声。我想我一定是足够幸运，能够进入到中南大学的信安专业，遇到这些可亲可爱的恩师与同伴。这四年里，我学习了信息安全充满魅力的专业知识，结识了很多良师益友，收获了充实又独一无二的一段时光。

毕业设计的过程中，在学习知识与提升自身能力之外，也遇到了很多问题与困难，是老师、同学与学校给予的支持和帮助，让我走到了如今圆满的一步。感恩感谢感激之情太多太多，诉说不尽。感谢学校提供的良好的学习环境与学习平台，开阔了我的视野，增长了我的见识；感谢在本科期间教授过我的老师，您们的无论是妙趣横生的知识又或是可爱的品格，都令我受益良多；感谢程序设计竞赛校队，教练与小伙伴、队友们在我算法竞赛的这段旅程中给予我了很多帮助；感谢极光网安实验室的学长学姐与小伙伴们，让我明晰了前进的方向，慢慢走进安全这个广阔深邃的宇宙，特别要感谢宋老师对实验室的关怀与付出；感谢我的室友和我的朋友对我学习和生活上的支持，与我一起分享快乐，为我排解忧虑；感谢傅建明老师与师兄师姐在毕业设计期间给予我的指导，帮助我解决了研究中的疑惑；感谢我的父母一直以来的鼓励与支持，他们并不过多干涉，但隔着一千多公里我也能感受到他们温柔的凝望。

特别感谢我的毕业设计导师段桂华老师，段老师有着温柔又智慧的力量，给予了我耐心又细心的教导和积极的鼓励；特别感谢一起相伴做毕设的朋友们，在我陷入困境时倾听我的困难与烦恼，帮我梳理问题所在，给予我有力的支持。

另外感谢网络，虽然我讨厌网络，但不可否认是网络的发展与素昧平生、优秀又愿意分享经验的陌生人，让我学到了精妙的知识与技术，读到了优雅的论文，促成了毕业设计的进展，甚至让我有所体悟。

最后以我并不完美的毕设，献给大学的末尾。衷心祝愿大家万事如意。

# 参考文献

1. An Exploration of Mobile First AI [EB/OL]. https://medium.com/swlh/an-exploration -of- mobile-first-ai-576c944efd36. 2018.
2. Huawei: AI安全白皮书 [EB/OL]. https://www-file.huawei.com/-/media/corporate/pdf /cyber-security/ai-security-white-paper-cn.pdf. 2018.
3. Artificial Intelligence Next Key Growth Area for Smartphones as Numbers Top Six Billion by 2020, IHS Markit Says [EB/OL]. https://news.ihsmarkit.com/press-release/ technology/artificial-intelligence-next-key-growth-area-smartphones-numbers-top-six-bi. 2017.
4. Over Half of Smartphone Owners Use Voice Assistants [EB/OL]. https://voicebot.ai/2018 /04/03/over-half-of-smartphone-owners-use-voice-assistants-siri-leads-the-pack/. 2018.
5. Mengwei Xu, Jiawei Liu, Yuanqiang Liu, et al. A First Look at DeepLearning Apps on Smartphones. The World Wide Web Conference on -WWW'19, 2019:2125-2136.
6. 纪守领, 杜天宇, 李进锋, 等. 机器学习模型安全与隐私研究综述 [EB/OL]. 软件学报. doi: 10.13328/j.cnki.jos.006131.
7. Converting model to C++ code [EB/OL]. https://mace.readthedocs.io/en/latest/user \_guide/advanced \_usage.html.
8. Strip visible string in ncnn [EB/OL]. https://github.com/Tencent/ncnn/wiki.
9. Sun Z, Sun R, Lu L. Mind Your Weight(s): A Large-scale Study on Insufficient Machine Learning Model Protection in Mobile Apps[J]. Arxiv, 2020:1-17. https://arxiv.org/abs/ 2002.07687.
10. Entropy(informationtheory) [EB/OL]. https://en.wikipedia.org/wiki/Entropy\_(informa tion\_theory)#Entropy\_as\_information\_content.
11. Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, Michael Wellman. Towards the Science of Security and Privacy in Machine Learning[C]. arXiv preprint arXiv:1611. 03814, 2016.
12. vera. AI与安全: Attack AI 总述 [EB/OL]. https://www.anquanke.com/post/id/181878, 2019-07-15.
13. 肖奇学, 许伟林, 李康. 对深度学习的逃逸攻击--探究人工智能系统中的安全盲区 [EB/OL]. https://blogs.360.cn/post/evasion-attacks-on-ai-system.html, 2017-10-18.
14. Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples[J]. arXiv preprint arXiv:1412.6572, 2014.
15. vera. AI与安全: Attack AI 破坏模型完整性--数据投毒攻击 [EB/OL]. https://www. anquanke.com/post/id/ 205097, 2020-05-14.
16. SeedLab. Android Repackaging Attack Lab[EB/OL]. https://seedsecuritylabs.org/Labs\_ 20.04/Files/ Android\_Repackaging/Android\_Repackaging.pdf
17. Apktool[EB/OL]. https://ibotpeaches.github.io/Apktool/
18. Twrp[EB/OL]. https://dl.twrp.me/bullhead/
19. Xposed[EB/OL]. https://repo.xposed.info/
20. Dynamic instrumentation toolkit for developers, reverseengineers, and security researchers [EB/OL]. https://frida.re/.